aanvulling cursus fortran 77 voor prime-computers

ing. J.M. den HAAN

ACADEMIC SERVICE

AANVULLING
CURSUS FORTRAN 77
voor Prime-computers

aanvulling cursus fortran 77 voor prime-computers

ing. J.M. den HAAN

ACADEMIC SERVICE

CIP-GEGEVENS

Haan, J.M. den

Aanvulling cursus Fortran 77 : voor Prime computers /
J.M. den Haan. - Den Haag : Academic Service
Met index.
ISBN 90-6233-113-0
SISO 365.3 SVS 8.12.3 UDC 681.3.06
Trefw.: programmeren ; computers.

Uitgegeven door: Academic Service
Postbus 96996
2509 JJ Den Haag
Zetwerk: multitASk, Blaricum
Druk: Krips Repro Meppel
Bindwerk: Meeuwis, Amsterdam
Omslagontwerp: JAM Gauw
ISBN 90 6233 113 0

VOORWOORD

Dit boek is een aanvulling op het eerder verschenen leerboek *Cursus FORTRAN 77*. Het heeft tot doel de lezer over de drempel te helpen bij het verwerken van de in het leerboek behandelde stof op Prime computers, werkend onder het PRIMOS bedrijfssysteem. Tussen beide boeken bestaat dus een theorie/praktijk-verhouding.

De praktijkgerichtheid van deze aanvulling maakt enkele inconsequenties op het gebied van de terminologie min of meer onvermijdelijk. Waar blijf je bijvoorbeeld met een puristische term zoals *bestand* als de systeemprogrammatuur je voortdurend confronteert met de term *file*? We hebben er bij dit boekje voor gekozen –met een bloedend hart weliswaar– enig water bij de wijn te doen. In plaats van (of zelfs naast) *bestand* spreken we van *file* en in plaats van *tekenreeks* over *string* of gewoon *tekst* –om maar enkele voorbeelden te noemen.

Het inhoudelijke materiaal van dit boek is gebaseerd op een aangepaste versie van PRIMOS (19.2.1/HBO) die in opdracht van de Stichting Computerbeheer Hoger Beroepsonderwijs te Enschede is gemaakt en via die stichting wordt gedistribueerd. Er wordt vrijwel uitsluitend aandacht besteed aan de zogenaamde batchgewijze verwerking. Alle editor- en programmavoorbeelden zijn op een Prime 400 bij Prime Benelux BV te Zoetermeer onder het genoemde systeem getest. Daarbij werd gebruik gemaakt van release 19.2.3 van de Prime FORTRAN 77 compiler. Een woord van dank aan Prime-medewerkers Jaap Breetvelt en Raimond Nicodem voor hun voortreffelijke medewerking bij het testen en becommentariëren van het manuscript is op zijn plaats.

Dank ook aan Bert Boorsma van de Stichting Computerbeheer HBO te Enschede voor zijn adviezen, het beschikbaar stellen van documentatie en zijn opmerkingen naar aanleiding van het bestuderen van het manuscript; aan Jan-Jaap van Horssen voor het kritisch doornemen van delen van het manuscript; en aan Inge Geerling-Engelbarts voor de typografische verzorging.

Tot slot de welhaast tot cliché verworden, maar desondanks oprecht gemeende standaardzin: voor op- of aanmerkingen die bij een eventuele herdruk van dit boek tot verbeteringen kunnen leiden houden we ons aanbevolen!

Pijnacker, februari 1984 Jack den Haan

NOTATIE AFSPRAKEN

Bij het definiëren can commandosyntax worden in dit boek de volgende notatieconventies gebruikt.

1. Hoofdletters stellen vaste gegevens voor die letterlijk moeten worden overgenomen.

2. Kleine letters stellen gegevens voor die door de gebruiker zelf moeten worden ingevuld.

   illustratie punten 1 en 2:

commandosyntax	:	RUN pnaam
intoetsen bijv.	:	RUN VOORBEELD

3. Blokhaken worden gebruikt om facultatieve elementen in een commando weer te geven.

   voorbeeld:

   ```
   OPEN ([UNIT=]unitnr [,FILE=bnaam]
                       [,STATUS=stat] [,IOSTAT=ios])
   ```

   betekent dat de elementen UNIT=, FILE=bnaam, STATUS=stat en IOSTAT=ios (met bijbehorende komma's) weggelaten mogen worden. De eenvoudigste vorm van het commando is dus:

   ```
   OPEN (unitnr)
   ```

   Bij het achterwege laten van facultatieve elementen vult het systeem in sommige gevallen zelf een standaardwaarde in. Deze wordt *default*- of *verstekwaarde* genoemd.

4. Editcommando's (zie hoofdstuk 3 en appendix) kunnen dikwijls worden afgekort tot een of meer letters, waarbij verschillende afkortingen dezelfde betekenis kunnen hebben. A, APP en APPE, bijvoorbeeld, zijn alle toegestane afkortingen van het APPEND-commando. de 'minimale' vorm bestaat altijd uit de eerste letter, eventueel gevolgd door een of meer van de direct daarop volgende letters. Bij het gebruik van blokhaken in commandonamen (zie punt 3) wordt met de minimale afkorting rekening gehouden, bijvoorbeeld A[PPEND].

5. Accolades geven aan dat uit de desbetreffende elementen een keuze móet worden gemaakt.

voorbeeld:

$$\text{MODE} \begin{Bmatrix} \text{NUMBER} \\ \text{NNUMBER} \end{Bmatrix}$$

betekent dat men of MODE NUMBER of MODE NNUMBER moet kiezen.

6. Een serie van drie opeenvolgende puntjes (een zogenaamde *ellipsis*) betekent een herhaling van het voorgaande element.

voorbeeld:

PTABSET tab1, tab2, ..., tabn

betekent dat men zou kunnen intoetsen:

PTABSET 7, 10, 13, 16, 19

(aangenomen dat 'tabn' een geheel getal mag zijn).

INHOUD

1 DE TERMINAL

1.1 INLEIDING

De naam *terminal* betekent letterlijk: eindpunt. In de computerpraktijk bedoelen we met 'terminal' een apparaat waarmee de gebruiker onder meer teksten in een computer kan brengen en hem opdracht kan geven daarmee iets te doen. Zo'n terminal wordt ook wel *eind-* of *werkstation* genoemd.

Met een terminal kan de gebruiker de computer bijvoorbeeld opdracht geven:

- een programma te starten
- de ingevoerde tekst onder een gegeven naam in het achtergrondgeheugen op te slaan
- een eerder opgeborgen tekst af te drukken
- zo'n tekst als een FORTRAN-programma te beschouwen en als zodanig te verwerken.

Bij zulke opdrachten is sprake van communicatie. De (menselijke) gebruiker biedt de computer via de terminal gegevens aan (*invoer*). De computer kan daar op zijn beurt min of meer zinvol op reageren door informatie op diezelfde terminal te produceren (*uitvoer*). De terminal is dus in feite een gecombineerd invoer/uitvoerapparaat.

De invoer vindt plaats door middel van een *toetsenbord*, de uitvoer door middel van een *beeldscherm* of door middel van een *afdrukmechanisme* waarbij tekens op papier worden geproduceerd. Terminals met een afdrukmechanisme noemt men ook wel *hardcopy* terminals. Dat wil zeggen: de computeruitvoer is blijvend op papier beschikbaar en kan niet – zoals bij een beeldscherm – door nieuwe gegevens worden gewist. Het scherm of afdrukmechanisme heeft behalve bij de uitvoer overigens ook een functie bij de invoer, namelijk om ingevoerde tekst zichtbaar te maken. In het kader van deze 'Aanvulling' zullen we onder 'terminal' een beeldschermterminal verstaan.

Bij beeldbuisterminals kunnen scherm en toetsenbord in één behuizing zijn ondergebracht. Ze komen echter ook als afzonderlijke

apparaten voor, onderling verbonden door een 'krulsnoer'. De meeste beeldschermterminals die op de HBO Prime-computers zijn aangesloten (TeleVideo 950) hebben zo'n 'los' toetsenbord.

1.2 BEDIENING

De bediening van een terminal is in grote lijnen vergelijkbaar met die van een elektrische schrijfmachine. Behalve de gebruikelijke 'huis-tuin-en-keuken'-tekens zoals letters, cijfers en leestekens, kent het terminal-toetsenbord echter ook aan aantal bijzondere toetsen en toetscombinaties. De belangrijkste hiervan zullen we nu bespreken.

Bijzondere toetsen en toetscombinaties

SHIFT — Zolang deze toets ingedrukt is krijgen we met de lettertoetsen hoofdletters en met de overige toetsen de 'bovenste' tekens. Is SHIFT niet ingedrukt, dan krijgen we kleine letters (mits de toets ALPHA LOCK niet is ingedrukt) en de 'onderste' tekens.

ALPHA LOCK — Wordt deze toets ingedrukt, dan blijft hij ook na het loslaten ingedrukt. Pas als hij opnieuw wordt ingedrukt komt hij weer naar boven. Als ALPHA LOCK ingedrukt is produceren de lettertoetsen altijd hoofdletters, ongeacht SHIFT. De overige toetsen worden door ALPHA LOCK niet beïnvloed.

RETURN — Met deze toets wordt een regel afgesloten, dat wil zeggen ter verwerking aan de computer aangeboden. Daarbij wordt de terminal automatisch op een nieuwe regel ingesteld. In een regel die nog niet door RETURN is afgesloten kunnen met behulp van de BACKSPACE-toets of een KILL (zie verderop) correcties worden aangebracht.

BACKSPACE — Indrukken van deze toets verwijdert het laatst ingetypte teken. Met tweemaal BACKSPACE worden de laatste twee tekens verwijderd, enzovoort.

voorbeeld:

> Het achtereenvolgens intoetsen van SUBPRO←←←ROUTINE (waarin BACKSPACE met ← is aangegeven) komt uiteindelijk in de computer als SUBROUTINE.

CTRL-X (KILL) Hiermee bedoelen we een combinatie van twee toetsen: de CTRL- (= ConTRoL) en de X-toets. In plaats van de X kan ook het minteken worden gebruikt. Beide combinaties worden in de standaard HBO-versie van PRIMOS opgevat als zg. KILL-teken. Ten gevolge hiervan worden alle tekens van de onderhavige regel verwijderd. Door de systeembeheerder kan eventueel ook een ander teken als KILL zijn of worden gekozen. Tijdens het editten (zie hoofdstuk 3) kan men met behulp van het PS-commando nagaan welk teken als KILL is gedefinieerd.

CTRL-H (ERASE) Hiermee bedoelen we een combinatie van twee toetsen: de CTRL- (= control) en de H-toets. Terwijl de CTRL-toets ingedrukt is, wordt ook de H-toets ingedrukt. De control-H combinatie wordt door de computer opgevat als een zogenaamd ERASE-teken (to erase = wissen). CTRL-H heeft hetzelfde effect als een BACKSPACE: het vernietigt het voorafgaande teken.

Ook voor het ERASE-teken kan door de systeembeheerder een ander teken zijn of worden gekozen. Het ingestelde teken is door middel van het PS-commando tijdens het editten te achterhalen.

\ (tabulatieteken) Alleen aktueel tijdens het gebruik van de editor (zie hoofdstuk 3). Heeft tot gevolg dat bij het afdrukken de verdere tekst op de onderhavige regel wordt opgeschoven tot de eerstvolgende tabulatorstop. Meer dan één tabulatorteken is toegestaan.

voorbeeld:

```
IF (A .GE. 0.0) THEN
\\SOMX = SOMX + X(I,J)
\ELSE
\\SOMY = SOMY + Y(I,J)
ENDIF
```

resulteert in:

```
IF (A .GE. 0.0) THEN
            SOMX = SOMX + X(I,J)
      ELSE
            SOMY = SOMY + Y(I,J)
ENDIF
```

De 'tabs' staan bij het editten standaard op posities 6, 12 en 30. Met het TABSET-commando (zie appendix) kunnen deze instellingen worden gewijzigd.

BREAK — Hierdoor wordt de uitvoering van een 'lopend' programma afgebroken. Dit komt vooral van pas als een programma in een oneindige lus terecht is gekomen. Bij het afbreken verschijnt op het scherm de tekst QUIT.

CTRL-S — Hiermee bedoelen we een combinatie van twee toetsen: de CTRL- (= control) en de S-toets. Terwijl de CTRL-toets ingedrukt is, wordt ook de S-toets ingedrukt. Deze toetscombinatie heeft tot gevolg dat de uitvoer naar de terminal tijdelijk wordt onderbroken. Dit is vooral nuttig bij het verschijnen van grote hoeveelheden uitvoer op het scherm. Zonder een onderbrekingsmogelijkheid zou uitvoer al van het scherm kunnen verdwijnen zonder dat men het heeft kunnen lezen.

CTRL-Q — Met deze toetscombinatie kan men de met CTRL-S onderbroken schermuitvoer hervatten.

Line/local mode

Terminals kunnen in *line* en *local mode* werken. 'Line mode' (de normale wijze van terminal-gebruik) betekent dat de terminal in principe op een computer is aangesloten. Werken in 'local mode' betekent dat er geen aansluiting op de computer is. Een hardcopy-terminal, bijvoorbeeld, functioneert dan in feite als een gewone schrijfmachine. Met een schakelaar of eventueel een zogenaamde *setup* procedure kan van de ene naar de andere mode worden geschakeld.

De cursor

Na het aanzetten en op bedrijfstemperatuur komen van een beeldschermterminal verschijnt op het scherm een al dan niet knipperend lichtvlekje ter grootte van één tekenpositie. Dit is de zogenaamde *cursor*, ook wel loper genoemd. De cursor kan verschillende vormen hebben, maar is meestal een rechthoekig blokje of een onderstrepingsteken.

Normaliter geeft de cursor de positie op het scherm aan waar men 'bezig' is. Na het intoetsen van een willekeurig teken komt dit teken op de plaats van de cursor te staan. De cursor zelf gaat naar de eerstvolgende tekenpositie op dezelfde regel of, als die regel vol is, naar de eerste tekenpositie op de volgende regel.

De cursor wordt onder andere beïnvloed door de RETURN- en BACKSPACE-toetsen, en kan door vier speciale besturingstoetsen (de pijltjes op het toetsenbord) in verticale en horizontale richting worden verplaatst. Tenzij de terminal in zogenaamde *block mode* werkt heeft het gebruik van laatstgenoemde toetsen alleen lokale betekenis.

2 WERKEN MET PRIMOS

2.1 INLEIDING

In het eerste hoofdstuk hebben we gezien dat we met behulp van een terminal met een computer kunnen communiceren. Zinvol communiceren vereist echter enige intelligentie. (Ook) van de computer. Wat hem betreft is die intelligentie in eerste instantie aanwezig in de vorm van een *bedrijfssysteem*, ook wel *besturingssysteem* of, op zijn Engels, *operating system (OS)* genoemd. Het bedrijfssysteem van de Prime computer is PRIMOS (*Prim*e *o*perating *s*ystem).

2.2 DE TAKEN VAN EEN BEDRIJFSSYSTEEM

Een bedrijfssysteem heeft een regelende en besturende functie. Het zorgt er onder meer voor dat de vele programma's die tegelijkertijd in de machine aanwezig kunnen zijn ordelijk worden afgewerkt en op hun beurt de beschikking krijgen over rekenfaciliteiten, randapparatuur, enzovoort.

Daartoe beschikt het bedrijfssysteem over *systeemprogramma's*. Voorbeelden daarvan zijn programma's voor:

- het opbouwen van tekstfiles (*editors*)
- het vertalen van programma's geschreven in hogere talen zoals FORTRAN (*compilers*)
- het kopiëren of verwijderen van files (filebeheer- of *utility*-programma's
- het aansturen van randapparatuur (*I/O drivers*).

In tegenstelling tot *gebruikersprogrammatuur* of *applicatieprogrammatuur* – gericht op toepassingen buiten de computer – heeft systeemprogrammatuur dus een ondersteunende functie, gericht op het efficiënt functioneren en kunnen gebruiken van de computer zelf.

2.3 DE OK- EN ER-PROMPT

De in het bedrijfssysteem opgenomen systeemprogramma's worden geactiveerd door commando's (niet te verwarren met programma-opdrachten) die de gebruiker bijvoorbeeld met een terminal aanbiedt. De commando's worden eerst 'ontleed' door een zogenaamde *command processor*, die op zijn beurt in staat is andere programma's te activeren.

Bij de Prime laat dit systeemprogramma onder andere van zich blijken door de tekst OK of ER op het scherm te produceren. Zo'n tekst heet *prompt* (to prompt = voorzeggen, souffleren). Het verschijnen van een OK-prompt betekent dat de computer een voorgaand commando heeft geaccepteerd, en klaar staat voor een nieuw commando. De ER (afkorting van error = fout) prompt verschijnt als een voorgaand commando niet goed verwerkt kon worden, bijvoorbeeld ten gevolge van een spelfout. In beide gevallen verwacht de computer na het verschijnen van de prompt aktie van de gebruiker.

Bij het aanbieden van commando's na een OK- of ER-prompt werken we op PRIMOS commandoniveau. Later zullen we zien dat er ook op andere niveaus commando's gegeven kunnen worden, bijvoorbeeld op editorniveau. Zulke commando's werken niet op PRIMOS-niveau.

2.4 INLOGGEN

Onder *inloggen* verstaan we het tot stand brengen van een werkverbinding met de computer. Ook dit proces wordt door PRIMOS geregeld. Daarvoor zijn twee authorisatiegegevens nodig: een *account* en een *wachtwoord*.

Account

Een account is een aan de gebruiker toegekende naam, die bij de HBO-systemen uit maximaal zes tekens kan bestaan (bij standaard Prime-systemen kan dit aantal groter zijn). De eerste twee tekens stellen bij de HBO Prime-systemen doorgaans een klas, groep of project voor, de overige een studentidentificatie (andere indelingen zijn mogelijk).

voorbeeld:

KKSTUD

klas : KK
student : STUD

Accountnamen worden door docent of systeembeheerder vastgesteld. Ze vormen tevens een zogenaamd *u*ser *f*ile *d*irectory (UFD).

Wachtwoord

Een wachtwoord kan door de gebruiker zelf worden veranderd. Dit gebeurt op de volgende manier:

```
OK, CPW <oud wachtwoord>
New password? <nieuw wachtwoord>
Reenter new password for confirmation: <nieuw wachtwoord>
OK,
```

De tekst tussen < > geeft hierbij de respons van de gebruiker aan. Het nieuwe wachtwoord wordt pas geaccepteerd als het oude correct was ingevuld en het nieuwe twee keer correct is ingevoerd.

Wachtwoorden bestaan uit maximaal zes tekens, waarvan het eerste geen cijfer of leesteken mag zijn. Controlcombinaties (CTRL- plus tekentoets) zijn toegestaan, maar moeten in verband met ongewenste neveneffecten met beleid worden gekozen.

LOGIN-procedure

Met een juiste combinatie van account en wachtwoord kunnen we ons als legale gebruiker aan PRIMOS bekend maken. We geven daartoe het commando (1):

```
LOGIN  account
```

Als ons account bijvoorbeeld KKSTUD is, dan toetsen we dus in:

```
LOGIN KKSTUD
```

Uiteraard moet PRIMOS eerst door middel van een prompt te kennen hebben gegeven dat hij klaar staat voor een commando. Zo'n prompt kunnen we zonodig forceren door een willekeurig teken in te toetsen en dan RETURN te geven.

Het volledige vraag-en-antwoordspel ziet er als volgt uit (2):

```
X <RETURN>
LOGIN PLEASE
ER! LOGIN KKSTUD <RETURN>
PASSWORD? GEHEIM <RETURN>

ACCOUNT (USER n) LOGGED IN dag, datum, tijd
WELCOME TO PRIMOS VERSION nn.n.n/HBO.
LAST LOGIN dag, datum, tijd.

U HEBT NOG nnnn MINUTEN AANSLUITTYD OVER
U GEBRUIKT nnnn DISC RECORDS, TOEGESTAAN nnnn.
```

waarbij *n* een cijfer is.

1. Zie notatieafspraken aan het begin van deze aanvulling.
2. Invoer van de gebruiker is herkenbaar aan de <RETURN>.

Problemen bij inloggen

Inloggen kan om verschillende redenen mislukken. Enkele hiervan zijn:

- Spelfout(en) in commandonaam, account en/of wachtwoord.
- Wachtwoord vergeten.
- Account en wachtwoord horen niet bij elkaar. Remedie: controleer uw authorisatiegegevens, eventueel via de systeembeheerder.
- Terminal reageert in het geheel niet. Mogelijke oorzaken:
  - Geen netspanning. Schakelaar staat niet aan, of storing in elektriciteitsnet.
  - De terminal staat in local mode. Ingetypte tekst verschijnt in dit geval wel op scherm of papier, maar de computer reageert er niet op. Remedie: zet de terminal via de zogenaamde setup-procedure in LINE-mode.
  - Bepaalde terminalkarakteristieken (bijvoorbeeld de zogenaamde baudrate) zijn niet goed ingesteld. Raadpleeg uw systeembeheerder.
  - De lijnverbinding tussen de terminal en de computer is verbroken.
  - Toevallige transmissiefout. Remedie: sla twee of drie willekeurige toetsen aan, geef RETURN en probeer opnieuw.
  - De computer is *down* (niet beschikbaar).
  - De terminal is defect.

2.5 ENKELE PRIMOS-COMMANDO'S

Heeft PRIMOS u eenmaal als legale gebruiker geaccepteerd, dan staan de 'normale' faciliteiten van het computersysteem tot uw beschikking. Enkele commando's die u in dit stadium kunt geven zijn:

- ED of ED xyz. Hierdoor wordt de *editor*, een permanent in de computer aanwezig (ssyteem)programma voor het inbrengen, opbergen en wijzigen van tekst (zie hoofdstuk 3), opgeroepen.

- SLIST xyz. Dit commando laat de inhoud van file (bestand) xyz op de terminal zien.

- SPOOL xyz. Drukt de inhoud van file xyz op de regeldrukker af.

- DELETE xyz. Verwijdert file xyz uit het (achtergrond)geheugen.
- LISTF. Geeft een lijst van de namen van de files die onder het ingelogde account geregistreerd staan.
- CPW. Verandert het wachtwoord.

2.6 FILENAAMCONVENTIES

Een *file* (ook wel *bestand* genoemd (1)) is een verzameling bij elkaar behorende gegevens die in een bepaalde vorm in het geheugen van een computer wordt opgeslagen, en door de computergebruiker als een eenheid kan worden 'aangesproken'. De naam waaronder dat 'aanspreken' plaatsvindt moet aan de volgende voorschriften voldoen:

1. Filenamen mogen uit maximaal 32 tekens bestaan.
2. Filenamen kunnen uitsluitend uit de volgende tekens worden opgebouwd:
   - hoofdletters en kleine letters A t/m Z
   - cijfers 0 t/m 9
   - de tekens & - $. _ (underscore) / en #
3. Het eerste teken van een filenaam mag elk van de onder punt 2 genoemde tekens zijn behalve een cijfer.
4. Spaties zijn in filenamen niet toegestaan.

PRIMOS maakt geen onderscheid tussen hoofdletters en kleine letters – de laatste worden intern geconverteerd naar hoofdletters.

voorbeelden:

```
A
a001
TEST
PROG1.FTN             legaal
$MAAND-RAPPORT
Pietje_Puk


A?
1234a                 illegaal
%ABC
Pietje Puk
```

1. Zo ook in het boek *Cursus FORTRAN 77* waarop dit boekje een aanvulling is. In verband met de terminologie waarmee men door PRIMOS en de bijbehorende systeemdocumentatie wordt geconfronteerd, gebruiken we hier voornamelijk de term 'file'.

2.7 VERBINDING VERBREKEN (UITLOGGEN)

De werkverbinding met de computer verbreekt men met het commando LOGOUT, eventueel af te korten tot LO. Naar aanleiding hiervan drukt PRIMOS de volgende boodschap op de terminal af:

```
account (USER n) LOGGED OUT dag, datum, tijd
TIME USED: nnH nnM CONNECT, nnM nnS CPU, nnM nnS I/O.
U HEBT NOG nnnn MINUTEN AANSLUITTYD OVER
U GEBRUIKT nnnn DISC RECORDS, TOEGESTAAN nnnn.
```

waarbij *n* een cijfer voorstelt.

De terminalsessie wordt hierbij dus beëindigd. Alle informatie die men tijdens de sessie heeft opgebouwd gaat daarbij verloren, tenzij men die in permanente files heeft ondergebracht (zie § 3.1.6).

3 DE EDITOR

3.0 INLEIDING

Onder *editor* verstaan we een hulpprogramma waarmee tekstfiles (bijvoorbeeld programmateksten of -gegevens) kunnen worden opgebouwd en/of gemodificeerd. In dit hoofdstuk leren we met zo'n editor om te gaan.

We doen dit aan de hand van een tweetal *edit sessies*, dat wil zeggen praktijksituaties waarin met concrete tekstfiles wordt gewerkt. Aanbevolen wordt deze sessies met behulp van een terminal op de voet te volgen.

Bij de behandeling van de editsessies komen de belangrijkste basiscommando's van de editor uitgebreid aan de orde. Het voert echter te ver om alle editcommando's op deze wijze te behandelen. Voor meer informatie zult u andere bronnen moeten aanboren, bijvoorbeeld de systeemdocumentatie van de fabrikant (1). Om daarbij gerichter te kunnen zoeken, en ook voor naslagdoeleinden, wordt in het appendix een overzicht van alle editcommando's gegeven.

3.1 EERSTE EDITSESSIE

3.1.0 Inleiding

In deze sessie maken we een tekstfile met als voorlopige inhoud:

```
AAAAAAA
BBBBBBB
CCCCCCC
XXXXXXX
YYYYYYY
ZZZZZZZ
```

---

1. *New User's Guide to EDITOR and RUNOFF*, nr. FDR 3104-101B.

Niet zo'n inspirerend stuk proza – toegegeven – maar wel 'simplisties' en daardoor gemakkelijk te hanteren.

Aan de hand van dit voorbeeld worden de volgende onderwerpen behandeld:

- het activeren van de editor
- het begrip *werkfile*
- het onderscheid tussen *input-* en *editmode*.

Daarna breiden we de file uit door:

- na CCCCCCC de regels DDDDDDD en EEEEEEE tussen te voegen
- na ZZZZZZZ de regels 0000000, 1111111 en 2222222 toe voegen.

Daarbij komen aan de orde:

- het begrip *pointer*
- het geheel of gedeeltelijk afdrukken van de werkfile.

Tenslotte laten we zien hoe de editsessie zo kan worden afgesloten dat de aangemaakte tekst als permanente file in het achtergrondgeheugen kan worden opgeslagen.

3.1.1 Activeren van de editor

De editor wordt geactiveerd door het ED-commando (1):

ED [filenaam]

De filenaam wordt uitsluitend gespecificeerd als het om een bestaande file gaat. Voor deze editsessie is dit niet het geval. We toetsen dus in:

```
ED
```

gevolgd door RETURN. De computer reageert nu met:

```
INPUT
```

waarbij de cursor op de eerste positie van de daaropvolgende regel blijft staan. De editor is nu in zogenaamde *inputmode*. Dit betekent dat alle tekst die nu wordt ingetoetst terechtkomt in een tijdelijk werkgebied in het interne geheugen: de zogenaamde *werkfile*. Uiteraard moeten alle regels met <RETURN> worden afgesloten.

We kunnen dus nu onze tekstregels intoetsen. De terminal-'dialoog' ziet er als volgt uit (de door de gebruiker in te toetsen tekst is onderstreept):

1. Zie de notatieafspraken aan het begin van deze 'Aanvulling'.

```
              commentaar

OK, ED        aktiveren van de editor
INPUT         editor klaar voor invoer
AAAAAAA       tekstregel 1
BBBBBBB       tekstregel 2
CCCCCCC       tekstregel 3
XXXXXXX       tekstregel 4
YYYYYYY       tekstregel 5
ZZZZZZZ       tekstregel 6
[]            cursor blijft staan
```

3.1.2 Pointercommando's

Bij het opbouwen van een tekstfile maakt de editor intern gebruik van een zogenaamde *pointer* (to point = (aan)wijzen). Deze wijst steeds de regel aan waarmee de editor op een gegeven moment bezig is. In het bestek van deze handleiding noemen we deze regel de *actuele regel* (Engels: current line).

In feite is de pointer niets anders dan een geheugenlokatie waarin door de editor automatisch een regelnummer wordt opgeslagen: aan het begin van de editsessie 0, na het invoeren van de eerste regel 1, enzovoort. De pointerwaarde kan worden veranderd of opgevraagd door een aantal commando's. Deze zijn samengevat in de volgende tabel.

tabel 3.1 : Overzicht pointercommando's

commando	afkorting	effect
TOP	T	Stelt de pointer in op de zogenaamde nulregel, d.w.z. juist voor de eerste tekstregel.
BOTTOM	B	Stelt de pointer in op de laatste tekstregel
NEXT [[±]n]	N	Pointer gaat n regels verder (n>0) of n regels terug (n<0), waarbij de nieuwe actuele regel wordt afgedrukt. Verstekwaarde n = 1.
POINT n	PO	Stelt de pointer in op regel n (komt overeen met TOP, NEXT n). Voorwaarde: n > 0.
WHERE	W	Geeft het nummer van de actuele regel. Vooral handig in combinatie met POINT en/of genummerde listings.

n = een of meer cijfers

Recente versies van de PRIME-editor kennen nog een andere mogelijkheid om regels in te stellen: het MODE INFO commando. Na het geven van dit commando kan men de werkfile regel voor regel op het scherm vertonen door eenvoudig op de RETURN/ENTER-toets te drukken. Dit vooral handig bij het 'scannen' van kortere stukken tekst. Uiteraard wordt de pointerwaarde bij iedere RETURN/ENTER gewijzigd.

In ons geval wijst de pointer na het invoeren van alle tekstregels naar regel 6. De volgende stap is het invoegen van de regels DDDDDDD en EEEEEEE na de regel CCCCCCC. Daarbij moet de pointer regel 3 aangeven. Dit kunnen we bijvoorbeeld bereiken met het commando NEXT -3, dat wil zeggen: zet de pointer drie regels terug (van 6 naar 3). Voordat we dit daadwerkelijk uitvoeren, behandelen we in de volgende paragraaf eerst het verschil tussen input- en editmode.

3.1.3 Overgang input- naar editmode en omgekeerd

Tijdens het invoeren van tekstregels (§ 3.1.1) stond de editor in zogenaamde *inputmode*. Dat wil zeggen alle ingevoerde tekst wordt geaccepteerd als inhoud van de werkfile, en dus ook van de uiteindelijk aan te maken tekstfile. Dit geldt ook voor eventuele commando's die we in dit stadium zouden invoeren. Het commando NEXT -3 (hoewel op zich correct) zou dus niet als zodanig worden herkend.

Om de editor duidelijk te maken dat het om een commando gaat in plaats van gewone tekst, moeten we eerst overschakelen naar *editmode* (ook wel *commandomode* genoemd). Dat kan eenvoudig door opnieuw (dat wil zeggen voor de tweede keer na het afsluiten van de laatste tekstregel) RETURN te geven. Voor het omgekeerde – dus van edit- overgaan naar inputmode – geldt precies hetzelfde. Ook hier zorgen twee opeenvolgende RETURNs voor het omschakelen van de ene mode naar de andere.

We kunnen nu de gewenste regels tussenvoegen. De terminaldialoog ziet er als volgt uit:

	commentaar
<RETURN>	druk op RETURN of ENTRY toets om van inputmode over te gaan naar editmode editor geeft op scherm een lege regel plus melding "EDIT"
EDIT	
N -3	3 regels terugspringen
CCCCCCC	editor laat de aktuele regel zien
<RETURN>	druk op RETURN/ENTRY voor overgang edit- naar inputmode editor geeft op scherm lege regel plus melding:
INPUT	
DDDDDDD	voer nieuwe tekstregel in
EEEEEEE	idem
[]	editor nog steeds in inputmode, cursor aan begin volgende regel

Het toevoegen van de regels 0000000, 1111111 en 2222222 (zoals gesteld in § 3.1.0) kan op dezelfde wijze plaatsvinden. De pointer moet nu echter naar de laatste regel wijzen. Behalve met het inmiddels bekende NEXT kan dit ook met het BOTTOM-commando:

	commentaar
<RETURN>	overgang input- naar editmode
	editorrespons
EDIT	idem
B	pointer op einde werkfile met BOTTOM
<RETURN>	overgang edit- naar inputmode
	editorrespons
INPUT	idem
0000000	invoer tekstregel
1111111	idem
2222222	idem
[]	cursor

3.1.4 Het INSERT-commando

Tot nu toe zijn tekstregels uitsluitend in de werkfile opgenomen tijdens inputmode. Er bestaat echter ook een INSERT-commando, waarmee tijdens editmode regels stuk voor stuk in de werkfile kunnen worden opgenomen. De vorm van het commando is:

I[NSERT] tekst

Tussen beide delen van het commando behoort één spatie te staan. Alle tekens na die spatie worden geacht deel uit te maken van de in te voeren tekstregel. Die tekstregel wordt na de actuele regel ingevoerd.

Een alternatief voor het laatste stukje terminaldialoog van § 3.1.3. is dus:

	commentaar
<RETURN>	druk op RETURN/ENTER toets
	editorrepons
EDIT	idem
B	pointer naar einde werkfile met BOTTOM
I 0000000	invoer tekstregel met INSERT
I 1111111	idem
I 2222222	idem

3.1.5 Afdrukken (PRINT)

Tijdens het editten kan het nuttig zijn de werkfile geheel of gedeeltelijk op het scherm te vertonen. Daarvoor is een PRINT-commando beschikbaar:

PRINT n

waarbij *n* het aantal af te drukken regels is.

De PRINT werkt vanaf de actuele regel. Willen we een overzicht van de gehele werkfile, dan moet de pointer dus eerst met (bijvoorbeeld) TOP op het begin van de werkfile worden ingesteld. De schermuitvoer kan desgewenst worden onderbroken en hervat met CTRL S respectievelijk CTRL Q. Na afloop van PRINT wijst de pointer naar de laatste afgedrukte regel.

In het volgende stukje dialoog wordt de gehele werkfile zoals hij er nu bijstaat afgedrukt.

```
                         commentaar

<RETURN>                 druk op RETURN/ENTER
                         editorrespons
EDIT                     idem
T                        pointer naar begin werkfile met TOP
P 23                     druk 23 regels af ( = 1 scherm)
.NULL.                   editormelding: begin werkfile
AAAAAAA                  eerste regel van de werkfile
BBBBBBB                  tweede regel van de werkfile
CCCCCCC                  enz.
DDDDDDD
EEEEEEE
XXXXXXX
YYYYYYY
ZZZZZZZ
0000000
1111111
2222222                  laatste regel van de werkfile
BOTTOM                   editormelding: einde werkfile
```

toetsvraag:

> Zoals in hoofdstuk 2 werd vermeld kan ook het SLIST-commando worden gebruikt om de inhoud van een file af te drukken. Kan dit commando tijdens een editsessie als alternatief voor PRINT worden gebruikt? Waarom wel/niet?

3.1.6 Afsluiten van de editsessie (FILE, QUIT)

Afsluiten van een editsessie houdt in dat het lopende editorprogramma wordt beëindigd, en dat we weer rechtstreeks met PRIMOS te maken krijgen. Daarbij moet een beslissing worden genomen over de werkfile die tijdens het editten is ontstaan.

Gewoonlijk zullen we deze willen bewaren. Er moet dan een permanente versie worden gemaakt met het commando:

FILE [filenaam]

De filenaam mag alleen worden weggelaten bij het editten van een bestaande file. In ons geval moet hij dus wel worden gespecificeerd. Gebeurt dit niet, dan zal de editor ons er naar vragen.

voorbeeld:

```
FILE
FILE NAME MUST BE SPECIFIED
?
FILE TEKST1
OK,
```

toelichting:

De inhoud van de werkfile wordt onder de naam TEKST1 in het achtergrondgeheugen opgeborgen. De editsessie wordt beëindigd. Dat PRIMOS daarna de teugels weer in handen neemt is te zien aan de OK-prompt.

Bij het kiezen van filenamen is het overigens oppassen geblazen. Als in bovenstaand voorbeeld de file TEKST1 al zou hebben bestaan, dan wordt hij als gevolg van het FILE-commando zonder waarschuwing overschreven!

Het QUIT-commando

Het is ook mogelijk een editsessie te beëindigen zonder de werkfile te bewaren. In dit geval geeft men in plaats van FILE het commando QUIT (eventueel af te korten tot Q). Het commando heeft geen parameters. QUIT is vooral nuttig als men in een bestaande file wijzigingen heeft aangebracht die bij nader inzien onterecht blijken. Door met QUIT af te sluiten wordt de situatie vóór het editten hersteld, en krijgt men dus de oorspronkelijke file terug.

In ons geval zou de editor na QUIT reageren met de melding FILE MODIFIED, OK TO QUIT? Dat wil zeggen hij heeft geconstateerd dat de file nieuw is (of dat een bestaande file is gemodificeerd), en dat er niet geFILEd is. Om vergissingen te voorkomen wordt er netjes geïnformeerd of de werkfile inderdaad mag verdwijnen. Antwoorden we op deze vraag met YES, YE, Y, O, OK of <RETURN> dan wordt de werkfile niet bewaard. Een willekeurig ander antwoord resulteert in de melding PLEASE FILE, dat wil zeggen een verzoek om alsnog te FILEn.

3.1.7 NULL regels

Bij het afdrukken van een werkfile met PRINT blijkt vóór de eigenlijke inhoud van de file een regel te staan met de tekst .NULL.. Ook aan het einde van de werkfile staat zo'n regel (te zien door BOTTOM gevolgd door PRINT te geven). Dit is een loze of 'dummy' regel die niet werkelijk bestaat. Hij wordt door de editor gebruikt om invoegen aan het begin mogelijk te maken, of om het einde van een file aan te geven.

3.1.8 Totaaloverzicht

We geven tenslotte een totaaloverzicht van de gehele editsessie.

```
                commentaar

OK, ED          aktiveren van de editor
INPUT           editormelding: inputmode ( = klaar voor invoer)
-------------------------------------------------------------------
AAAAAAA         invoeren eerste deel van de tekst
BBBBBBB
CCCCCCC
XXXXXXX
YYYYYYY
ZZZZZZZ
-------------------------------------------------------------------
<RETURN>        RETURN/ENTER voor overgang input- naar editmode
                editorrespons
EDIT            idem
N -3            pointer naar regel CCCCCCC
<RETURN>        RETURN/ENTER voor overgang edit- naar inputmode
                editorrespons
INPUT           idem
DDDDDDD         invoer nieuwe tekstregel
EEEEEEE         idem
-------------------------------------------------------------------
<RETURN>        RETURN/ENTER voor overgang input- naar editmode
B               pointer naar einde werkfile met BOTTOM
<RETURN>        RETURN/ENTER voor overgang edit- naar inputmode
0000000         invoer nieuwe tekstregel
1111111         idem
2222222         idem
-------------------------------------------------------------------
<RETURN>        RETURN/ENTER voor overgang input- naar editmode
T               pointer naar begin werkfile met TOP
P 23            druk 23 regels af ( = 1 scherm)
.NULL.          editormelding: begin werkfile
AAAAAAA         eerste tekstregel
BBBBBBB         tweede tekstregel
CCCCCCC         enz.
DDDDDDD
EEEEEEE
XXXXXXX
YYYYYYY
ZZZZZZZ
0000000
1111111
2222222         laatste tekstregel
BOTTOM          editormelding: einde werkfile
-------------------------------------------------------------------
FILE TEKST1     opslag werkfile in achtergrondgeheugen en
                editor verlaten
OK, []          terug op PRIMOS commando niveau
```

3.2 TWEEDE EDITSESSIE

3.2.0 Doelstellingen

Uitgaande van de in de vorige editsessie opgebouwde tekstfile (TEKST1) behandelen we nu:

- het activeren van de editor voor een bestaande file
- het opzoeken van een tekstdeel
- het wijzigen van een tekstregel
- het verwijderen van tekstregels
- het toevoegen aan een tekstregel.

Daarbij ontstaat een gewijzigde werkfile die we onder de naam TEKST2 in het achtergrondgeheugen opslaan. De oorspronkelijk file blijft onder de naam TEKST1 voortbestaan.

3.2.1 Activeren van de editor

We activeren de editor met het inmiddels bekende commando:

ED [filenaam]

De bedoeling is TEKST 1 (de in de vorige editsessie aangemaakte tekstfile) verder te bewerken. Omdat het in tegenstelling tot de vorige editsessie om een bestaande file gaat, moeten we nu wel een filenaam specificeren, dus:

```
OK, ED TEKST1 <RETURN>
```

Op het scherm verschijnt nu de melding EDIT, ten teken dat de opgegeven file aangesloten is en bewerkt kan worden.

toetsvraag:

> Deze reactie is anders dan bij het activeren van de editor zonder filenaam (zie eerste editsessie). Waarom?

Voor alle duidelijkheid laten we de file met behulp van PRINT op het scherm afdrukken:

```
                     commentaar

OK, ED TEKST1        editor aktiveren
EDIT                 editorrespons
P 23                 druk 23 regels af ( = 1 scherm)
.NULL.               editormelding: begin werkfile
AAAAAAA              eerste tekstregel in werkfile
BBBBBBB              tweede tekstregel
CCCCCCC              enz.
DDDDDDD
XXXXXXX
YYYYYYY
ZZZZZZZ
0000000
1111111
2222222              laatste tekstregel
BOTTOM               editormelding: einde werkfile
```

3.2.2 Opzoeken van tekst (LOCATE en FIND)

Stel dat we de regels XXXXXXX t/m ZZZZZZZ willen vervangen door FFFFFFF, GGGGGGG en HHHHHHH. Uiteraard zal de pointer – die nu 'onderaan' staat – bij het vervangen van een regel naar die regel moeten wijzen. We kennen inmiddels een commando om dit te verwezenlijken: NEXT -n. We kunnen de editor ook opdracht geven rechtstreeks een stuk tekst op te zoeken. Dat gaat met de commando's LOCATE en FIND.

Bij beide commando's wordt de op te zoeken tekst in de commandoregel opgenomen. Bij LOCATE wordt de gehele werkfile vanaf de actuele regel doorzocht op het voorkomen van die tekst (die we gemakshalve *doeltekst* zullen noemen). Wordt de doeltekst gevonden, dan wordt de regel waarin deze voorkomt in zijn geheel op het scherm afgedrukt. De pointer – die tijdens het zoekproces steeds wordt verhoogd – blijft deze regel aanwijzen. Wordt de doeltekst niet gevonden, dan verschijnt de melding BOTTOM. De pointer wijst dan de laatste regel van de werkfile aan.

De vorm van het LOCATE-commando is:

L[OCATE] doeltekst

Het FIND-commando werkt op soortgelijke wijze, met dien verstande dat de doeltekst geacht wordt in de eerste positie van een regel te beginnen. De vorm is:

F[IND] doeltekst

Voor beide commando's geldt:

- na de naam van het commando (LOCATE of FIND, eventueel af te korten tot L respectievelijk F) moet één spatie staan
- de ingetoetste tekens na de spatie vormen de doeltekst
- de doeltekst mag geen KILL- of een semico-teken (|, zie p.51) bevatten.

Toegepast op TEKST1 in de toestand waarin hij zich na de laatste (PRINT-) bewerking bevindt:

```
                commentaar

T               pointer naar begin werkfile met TOP
F X             zoek regel die met X begint
XXXXXXX         regel gevonden
[]
```

toetsvragen:

1. Zou het LOCATE-commando in dit geval bruikbaar zijn geweest? Zo ja, hoe?

2. Stel dat alle regels in de werkfile eindigen met X, dus:

   AAAAAAAX
   BBBBBBBX
   enz.

   De pointer wijst de eerste regel aan. Welk effect heeft dan het commando FIND X? Welk effect zou in plaats daarvan het commando LOCATE X hebben gehad? En het commando LOCATE XX (wederom in plaats van FIND X)?

3.2.3 Het RETYPE-commando

Met dit commando kan de actuele regel in zijn geheel worden vervangen. De vorm van het commando is:

R[ETYPE] tekst

Ook hier moet er een spatie staan tussen naam- en tekstdeel van het commando. Eventueel volgende spaties mogen worden geacht deel uit te maken van de nieuwe tekstregel. De pointer wordt bij RETYPE niet veranderd en wijst na afloop dus de nieuwe regel aan.

Toegepast op ons probleem (het vervangen van de regels XXXXXXX t/m ZZZZZZZ door de regels FFFFFFF, GGGGGGG en HHHHHHH):

```
                  commentaar

R FFFFFFF         vervang aktuele regel (XXXXXXX) door FFFFFFF
N                 stel volgende regel in
YYYYYYY           automatische controleafdruk nieuwe regel
R GGGGGGG         vervang door GGGGGGG
N                 stel volgende regel in
ZZZZZZZ           automatische controleafdruk nieuwe regel
R HHHHHHH         vervang door HHHHHHH
T.I P 23          druk eerste 23 regels af
.NULL.            editormelding: begin werkfile
AAAAAAA           eerste tekstregel
BBBBBBB           tweede tekstregel
CCCCCCC           enz.
DDDDDDD
EEEEEEE
FFFFFFF
GGGGGGG
HHHHHHH
0000000
1111111
2222222           laatste tekstregel
BOTTOM            editormelding: einde werkfile
[]
```

N.B.: Normaal gesproken kunnen meerdere commando's op één regel worden gegeven als ze door een verticale streep (het semicoteken, zie p.51) zijn gescheiden.

3.2.4 Het CHANGE-commando

Dit commando dient om een tekstregel geheel of gedeeltelijk te vervangen door andere tekst. De vorm van het commando is:

C[HANGE] /tekst1/tekst2/ [G] [n]

Iets ingewikkelder, dus, dan we tot nu toe hebben gezien. We beginnen met de eenvoudigste vorm door de facultatieve delen weg te laten:

C /tekst1/tekst2/

Hierin is *tekst1* de tekst die door *tekst2* wordt vervangen. De schuine streep (slash) dient om de teksten af te bakenen. Mocht de slash zelf deel uitmaken van de tekst, dan kan in plaats daarvan een ander teken worden gebruikt. Het commando heeft in deze vorm uitsluitend betrekking op de actuele regel.

Een andere manier om de eerste wijziging van § 3.2.3 uit te voeren is dus:

```
C /XXXXXXX/FFFFFFF/
```

Uiteraard verdient het RETYPE-commando hier de voorkeur. Daarbij

behoeft immers alleen de nieuwe regelinhoud opgegeven te worden. CHANGE is pas zinvol als het gaat om het vervangen van een deel van een regel. Het vervangen wordt bij deze vorm van het commando overigens maar één keer uitgevoerd, ook als *tekst1* meerdere malen in de regel voorkomt.

General CHANGE

We nemen nu de toevoegingen [G] en [n] onder de loep. De 'G' is een afkorting van *general* (= algemeen). Hij verandert de 'eenmalige' CHANGE in een 'algemene' CHANGE, waarbij de gespecificeerde vervanging de gehele regel door wordt uitgevoerd.

voorbeeld:

> Uitvoering van het commando C /X/F/ G op de regel XXXXXXX levert FFFFFFF (in dit geval is CHANGE dus wel een redelijk alternatief voor RETYPE).

Met de toevoeging 'n' – waarbij n een geheel getal is dat bij voorkeur groter is dan 2 – kan worden aangegeven dat de vervangingsoperatie behalve op de actuele, ook op de eerstvolgende n-1 regels moet worden uitgevoerd. De pointer verandert daarbij van waarde.

3.2.5 Het DELETE-commando

Met dit commando kunnen één of meer regels uit de werkfile worden verwijderd. De vorm is:

```
D[ELETE] [n]
```

Ook hier is n weer een geheel getal. In n groter dan 1, dan worden de actuele en de daaropvolgende n-1 regels verwijderd. Voor negatieve n geldt hetzelfde, maar dan in omgekeerde richting. Is n=1, of wordt n weggelaten, dan verdwijnt alleen de actuele regel.

We passen een en ander toe op onze werkfile door de laatste drie regels (0000000, 1111111 en 2222222) te verwijderen.

```
                    commentaar

T | F 0000000       lokaliseer eerste te verwijderen regel
0000000             regel gevonden
D 3                 verwijder 3 regels
BOTTOM              editormelding: einde werkfile
T, P 23             druk eerste 23 regels af
.NULL.              editormelding: begin werkfile
AAAAAAA             eerste tekstregel
BBBBBBB             tweede tekstregel
CCCCCCC             enz.
DDDDDDD
EEEEEEE
FFFFFFF
GGGGGGG
HHHHHHH             laatste tekstregel
BOTTOM              editormelding: einde werkfile
[]
```

3.2.6 Het 'DELETE TO'-commando

De vorm van dit commando is:

D[ELETE] TO doeltekst

De werking is analoog aan de gewone DELETE (zie boven) met dien verstande dat, in plaats van een vast aantal regels, alle regels vanaf de actuele tot de regel waarin de doeltekst voorkomt worden verwijderd.

opmerkingen:

1. Er dient één spatie tussen TO en de doeltekst te staan.
2. Komt de doeltekst in of na de actuele regel niet voor, dan worden alle regels vanaf de actuele verwijderd. Een typefout in de doeltekst kan dus rampzalige gevolgen hebben!

3.2.7 Het APPEND-commando

Het Engelse werkwoord 'to append' betekent 'aanhechten' of 'bijvoegen'. Zoals de naam suggereert, kunnen we met het APPEND-commando bestaande regels van een werkfile met extra tekst uitbreiden.

De vorm van het commando is:

A[PPEND] tekst

waarbij commandonaam en tekst door één spatie gescheiden dienen te worden.

voorbeeld:

Het commando APPENDbbMIES (waarbij b een spatie voorstelt) toegepast op de regel AAP NOOT levert AAP NOOT MIES. Het commando APPEND MIES zou in plaats daarvan AAP NOOTMIES geven.

Puntkomma's als onderdeel van de toe te voegen tekst zijn niet toegestaan. De nieuwe regel wordt ter controle afgedrukt.

We passen onze werkfile nu zo aan dat elke regel een tweecijferig volgnummer krijgt, dus AAAAAAA01, BBBBBBB02, enzovoort.

```
                    commentaar

T | N               lokaliseer eerste tekstregel
AAAAAAA             automatische controle-afdruk t.g.v. NEXT
A 01                regel d.m.v. APPEND uitbreiden met tekst 01
AAAAAAA01           automatische controle-afdruk t.g.v. APPEND
N                   stel volgende regel in
BBBBBBB             automatische controle-afdruk t.g.v. NEXT
A 02                regel uitbreiden met tekst 02
BBBBBBB02           automatische controle-afdruk t.g.v. APPEND
N                   enzovoort
CCCCCCC
A 03
CCCCCCC03
N
DDDDDDD
A 04
DDDDDDD04
N
EEEEEEE
A 05
EEEEEEE05
N
FFFFFFF
A 06
FFFFFFF06
N
GGGGGGG
A 07
GGGGGGG07
N
HHHHHHH
A 08
HHHHHHH08
T | P 23            controle afdruk maken van gehele werkfile
.NULL.              editormelding
AAAAAAA01           eerste tekstregel
BBBBBBB02           tweede tekstregel
CCCCCCC03           enz.
DDDDDDD04
EEEEEEE05
FFFFFFF06
GGGGGGG07
HHHHHHH08           laatste tekstregel
BOTTOM              editormelding: einde werkfile
[]
```

3.2.8 Afsluiten van de sessie

In deze editsessie zijn we uitgegaan van de reeds aangemaakte file TEKST1. Zoals eerder opgemerkt, is het de bedoeling dat deze file in zijn oorspronkelijke vorm blijft voortbestaan, terwijl de werkfile in zijn huidige vorm onder de naam TEKST2 in het achtergrondgeheugen wordt opgeslagen.

Dit kunnen we bereiken door een FILE-commando te geven met TEKST2 als filenaam. Dus:

```
FILE TEKST2 <RETURN>
```

Met LISTF kunnen we het een en ander desgewenst controleren.

3.2.9 Totaaloverzicht

We geven tenslotte een overzicht van de essentie van deze editsessie.

```
                  commentaar

OK, ED TEKST1     aktiveer editor voor bestaande file
EDIT              editorrespons
-----------------------------------------------------------------
P 23              druk eerste 23 regels af ( = 1 scherm)
.NULL.
AAAAAAA
BBBBBBB
CCCCCCC
DDDDDDD
EEEEEEE
FFFFFFF
GGGGGGG
HHHHHHH           laatste tekstregel
BOTTOM            editormelding: einde werkfile
-----------------------------------------------------------------
```

zie volgende bladzijde

```
T | N                lokaliseer eerste tekstregel
AAAAAAA              automatische controle-afdruk t.g.v. NEXT
A 01                 regel uitbreiden d.m.v. APPEND met tekst 01
AAAAAAA01            automatische controle-afdruk t.g.v. APPEND
N                    overige tekstregels op analoge wijze uitbreiden
BBBBBBB
A 02
BBBBBBB02
N
CCCCCCC
A 03
CCCCCCC03
N
DDDDDDD
A 04
DDDDDDD04
N
EEEEEEE
A 05
EEEEEEE05
N
FFFFFFF
A 06
FFFFFFF06
N
GGGGGGG
A 07
GGGGGGG07
N
HHHHHHH
A 08
HHHHHHH08
------------------------------------------------------------------
T | P 23             druk eerste 23 regels af
.NULL.
AAAAAAA01
BBBBBBB02
CCCCCCC03
DDDDDDD04
EEEEEEE05
FFFFFFF06
GGGGGGG07
HHHHHHH08            laatste tekstregel
BOTTOM               editormelding: einde werkfile
------------------------------------------------------------------
FILE TEKST2          werkfile opslaan in achtergrondgeheugen en
                     editor verlaten
OK, []
```

4 HET VERWERKEN VAN FORTRAN-PROGRAMMA'S

4.0 INLEIDING

In dit hoofdstuk komt het verwerken van FORTRAN-programma's aan de orde. Na een bespreking van de basiszaken – het compileren, en uitvoeren– wordt uitvoerig aandacht besteed aan de programma-in- en uitvoer. Een en ander wordt toegelicht aan de hand van een tweetal basisprogramma's, PROG1.F77 en PROG2.F77. Van het tweede basisprogramma worden verschillende versies behandeld (PROG2A.F77 t/m PROG2D.F77).

4.1 VOORBEELDPROGRAMMA

Als voorbeeld nemen we een programma dat de tweede macht van een reeks natuurlijke getallen bepaalt. Voor de uitvoer van de programmaresultaten maken we gebruik van de in *Cursus FORTRAN 77* veel toegepaste 'PRINT *'-opdracht. Bij de Prime 'schrijft' deze opdracht (normaal gesproken) naar de 'eigen' terminal, dat wil zeggen naar de terminal waarmee het programma werd gestart.

We voeren het programma met de editor in:

```
OK, ED
INPUT
C      DIT PROGRAMMA BEPAALT DE TWEEDE MACHT VAN DE GETALLEN
C      1 T/M 20, EN DRUKT DEZE OP DE EIGEN TERMINAL AF

       INTEGER GETAL, MACHT

       DO 10 GETAL = 1, 20
          MACHT = GETAL ** 2
          PRINT *, 'TWEEDE MACHT VAN ', GETAL, ' IS ', MACHT
10     CONTINUE

       STOP
       END
<RETURN>
EDIT
FILE PROG1.F77
OK, []
```

4.2 CONTROLEREN OP SYNTACTISCHE FOUTEN

We kunnen het programma van § 4.1 nu door de computer laten controleren op syntactische correctheid (syntax checking). Dit houdt in dat het programma door de compiler wordt verwerkt, echter zonder een definitief machinetaalprogramma te produceren. Bevat het programma syntactische fouten (dus fouten tegen de regels van FORTRAN) dan worden die door meldingen aangegeven.

De commandovorm is:

F77 *naam*

waarbij *naam* de naam is van de FORTRAN tekstfile, in ons geval dus PROG1.F77. Hierbij is F77 een standaard extensie die eventueel kan worden weggelaten. Dus:

```
OK, F77 PROG1 <RETURN>
```

Naar aanleiding hiervan verschijnt de volgende tekst op het scherm:

```
OK, F77 PROG1
[F77 Rev. 19.2.3]
0000 ERRORS [<.MAIN.> F77-REV 19.2.3]
OK,
```

Hierin is voor ons de derde regel de belangrijkste: 0000 ERRORS, dat wil zeggen er zijn geen syntactische fouten. De hier beschreven werkwijze is overigens interactief. Het alternatief is het batch-commando FT77. Na dit commando verschijnt een melding dat de 'job' is aangeboden, gevolgd door OK. Het scherm is dan vrij voor andere bezigheden. Afhankelijk van de drukte op het systeem verschijnen enige tijd later berichten die aangeven dat de job gestart en gestopt is. Men kan nu de zg. COMO-file bekijken (in ons geval met het commando SLIST PROG1.COMO) om eventuele fouten te achterhalen.

Nu het geval dat het programma wel syntactische fouten bevat. Als voorbeeld nemen we de volgende FORTRAN-tekst:

```
C     DIT PROGRAMMA BEPAALT DE TWEEDE MACHT VAN DE GETALLEN
C     1 T/M 20, EN DRUKT DEZE OP DE EIGEN TERMINAL AF

      INTEGER GETAL, MACHT

      DO 10 GETAL = 1, 20
         MACHT = GETAL ** 2
         PRINT *, 'TWEEDE MACHT VAN ', GETAL, ' IS ', MACHT
1O    CONTINUE

      STOP
      END
```

Hetzelfde programma als boven dus, maar nu – ter illustratie – met één syntactische fout: de letter O in plaats van het cijfer 0 in de label van de CONTINUE-opdracht. Dit levert na het FT77-commando de volgende uitvoer:

```
OK, FT77 PROG1A
[F77 Rev. 19.2.3]

ERROR 289 SEVERITY 3 BEGINNING ON LINE 9
Unexpected character appears in a label field. Only blanks and
digits are permitted.


ERROR 351 SEVERITY 3 BEGINNING ON LINE 12
Incorrect do-loop nesting. The statement label "10" does
not appear on a statement that follows the do statement
that references it or the do-loop is nested
incorrectly.


ERROR 342 SEVERITY 3 BEGINNING ON LINE 6
The statement label "10" has not been defined.

0003 ERRORS [<.MAIN.> F77-REV 19.2.3]
OK,
```

Een nogal uitgebreid verhaal, maar na de eerste schrik waarschijnlijk toch wel duidelijk!

4.3 HET MAKEN VAN EEN RUNFILE

Is een programma syntactisch in orde, dan kunnen we de computer er een zogenaamde *runfile* van laten maken, dat wil zeggen een kant-en-klaar uitvoerbaar programma. Het commando daarvoor heeft de volgende vorm:

FT77RUN naam

waarbij *naam* uiteraard de naam van de FORTRAN tekstfile voorstelt. Net als bij het FT77-commando kan de extensie F77 (met voorafgaande punt) achterwege blijven.

We passen een en ander toe op ons (correcte) demonstratieprogramma:

```
OK, FT77RUN PROG1
#
[JOB rev 19.0]
Your job, #00022, was submitted to queue QUEUE..
Home=<HBOSYS>AC>ACSERV
[BATCH rev 19.0]

Number of waiting and held jobs:

Queue  Jobs
------ ----
QUEUE.  1

No running jobs.
"
OK,
```

Hierbij is 00022 het volgnummer dat PRIMOS aan ons programma ('job') heeft toegekend, en <HBOSYS>AC>ACSERV de zogenaamde *home directory*. Beide gegevens zijn uiteraard variabel.

Behalve de runfile ontstaat ook een zogenaamde COMO-file, in ons geval PROG1.COMO. Hierin staan eventuele routines vermeld waaraan direct of indirect in uw programma wordt gerefereerd, maar die het systeem niet kan vinden of aansluiten. SLISTen van de COMO-file levert:

```
OK, SLIST PROG1.COMO
[SEG rev 19.2.2]
#    MAP PROG1.SEG 6

#    QUIT
* INDIEN ER NIET GEVONDEN ROUTINES ZIJN TUSSEN MAP EN QUIT
* DAN KONTAKT OPNEMEN MET UW DOCENT
OK,
```

Tussen MAP en QUIT staan in dit geval geen routines vermeld. De zaak is dus gezond en we kunnen het programma laten uitvoeren (zie volgende paragraaf).

4.4 HET UITVOEREN VAN EEN PROGRAMMA

Het programma is nu klaar om te worden uitgevoerd. Daartoe geven we een RUN-commando:

RUN programmanaam

Toegepast op ons programma:

```
OK, RUN PROG1 <RETURN>
```

Dit levert de volgende informatie op het beeldscherm van de terminal:

```
OK, RUN PROG1
  TWEEDE MACHT VAN          1 IS          1
  TWEEDE MACHT VAN          2 IS          4
  TWEEDE MACHT VAN          3 IS          9
  TWEEDE MACHT VAN          4 IS         16
  TWEEDE MACHT VAN          5 IS         25
  TWEEDE MACHT VAN          6 IS         36
  TWEEDE MACHT VAN          7 IS         49
  TWEEDE MACHT VAN          8 IS         64
  TWEEDE MACHT VAN          9 IS         81
  TWEEDE MACHT VAN         10 IS        100
  TWEEDE MACHT VAN         11 IS        121
  TWEEDE MACHT VAN         12 IS        144
  TWEEDE MACHT VAN         13 IS        169
  TWEEDE MACHT VAN         14 IS        196
  TWEEDE MACHT VAN         15 IS        225
```

```
  TWEEDE MACHT VAN            16 IS           256
  TWEEDE MACHT VAN            17 IS           289
  TWEEDE MACHT VAN            18 IS           324
  TWEEDE MACHT VAN            19 IS           361
  TWEEDE MACHT VAN            20 IS           400
 **** STOP
```

Zolang niet is uitgelogd blijft de runfile aanwezig. Hij kan dan eventueel opnieuw worden uitgevoerd.

4.5 DE 'READ *'-OPDRACHT

Voor het inlezen van gegevens wordt in *Cursus FORTRAN 77* doorgaans de tegenhanger van de 'PRINT *'-opdracht gebruikt: de READ *. Net als de PRINT * heeft deze opdracht bij de Prime ook betrekking op de terminal. We demonstreren dit aan de hand van het volgende programma (ontleend aan § 6.8 van *Cursus FORTRAN 77*):

```
C     DIT PROGRAMMA TELT HET AANTAL MENSEN IN DE LEEFTIJDSCATEGORIE
C     18 T/M 65 JAAR

      INTEGER MENS, LFTIJD

      READ *, LFTIJD
      MENS = 0

20    IF (LFTIJD .EQ. -1) GO TO 30
         IF (LFTIJD .GE. 18 .AND. LFTIJD .LE. 65) THEN
            MENS = MENS + 1
         END IF
         READ *, LFTIJD
         GOTO 20
30    CONTINUE

      PRINT *, 'AANTAL MENSEN 18 T/M 65 JAAR = ', MENS
      STOP
      END
```

We zullen dit programma PROG2.F77 noemen. Bij uitvoering ervan gebeurt ogenschijnlijk niets. De cursor springt naar het begin van de volgende regel en blijft daar staan. Op dat moment is de computer aangeland bij de eerste READ * -opdracht (althans bij de machine-instructies die daarmee overeenkomen). Het wachten is nu op het invoeren, door de gebruiker, van de eerste getalwaarde. Toetsen we nu een waarde in die ongelijk is aan -1 en geven we daarna RETURN of ENTER, dan zal de cursor naar de volgende regel springen en daar ten gevolge van de tweede READ opnieuw blijven staan. Het wachten is nu op de volgende getalwaarde.

Dit proces gaat door tot de afsluitwaarde -1 wordt ingevoerd. Naar aanleiding van de PRINT verschijnt dan het resultaat op het scherm en, ten gevolge van de STOP-opdracht, de melding "**** STOP".

De terminaldialoog (afgezien van het maken van de runfile, en gebaseerd op de invoerwaarden in *Cursus FORTRAN 77*) ziet er als volgt uit:

```
OK, RUN PROG2
16
25
31
12
28
69
-1
 AANTAL MENSEN 18 T/M 65 JAAR =              3
**** STOP

OK,
```

opdracht:

Ga dit na door het programma met de editor op te bouwen, en op de in 4.2 t/m 4.4 beschreven wijze te verwerken.

Gebruik van invoerprompts

Het bovenstaande programma is niet bepaald 'gebruikersvriendelijk' te noemen. De gebruiker krijgt immers geen enkele indicatie dat er van hem (of haar ...) actie wordt verlangd. We kunnen dit verbeteren door vóór de READ-opdrachten een PRINT te plaatsen, die voor een toelichtende tekst op het scherm zorgt. Bijvoorbeeld:

```
PRINT *, 'GEEF LEEFTIJD:
READ  *,  LEEFTD
```

Net als bij de OK- en ER-teksten van PRIMOS wordt zo'n tekst *prompt* of *invoerprompt* genoemd.

opdracht:

Pas PROG2.F77 zo aan dat bij uitvoering invoerprompts op het scherm verschijnen. Noem deze versie PROG2A.F77. Maak een runfile en laat het programma uitvoeren. Welk nadeel heeft het op deze wijze genereren van een prompt?

Advies: ga bij het editten uit van PROG2.F77, voer de noodzakelijke wijzigingen uit en verlaat de editor met het commando FILE PROG2A.F77.

Invoerprompt en -waarde op één regel

Dankzij een speciale bibliotheekroutine van Prime kan voorkomen worden dat een invoerprompt en de bijbehorende invoerwaarde op verschillende regels komen te staan. De routine (voluit: subroutine) heet TNOUA, en wordt via een CALL met twee parameters aangeroepen: een tekenreeks (string) en de lengte van die tekenreeks in zogenaamde short integer vorm (1). De short integer kan worden verkregen met behulp van de standaardfunctie INTS. Voorbeeld:

```
CALL TNOUA('DIT IS EEN PROMPT: ', INTS(19))
```

De routine moet met een zogenaamd EXTERNAL-statement worden gedeclareerd.

Toepassing op PROG2.F77 levert:

```
C     AANPASSING PROG2 MET INVOERPROMPTS D.M.V. ROUTINE TNOUA

      EXTERNAL TNOUA
      INTEGER MENS, LFTIJD

      CALL TNOUA('GEEF LEEFTIJD: ', INTS(15))
      READ *, LFTIJD
      MENS = 0

20    IF (LFTIJD .EQ. -1) GO TO 30
         IF (LFTIJD .GE. 18 .AND. LFTIJD .LE. 65) THEN
            MENS = MENS + 1
         END IF
         CALL TNOUA('GEEF LEEFTIJD: ', INTS(15))
         READ *, LFTIJD
         GOTO 20
30    CONTINUE

      PRINT *, 'AANTAL MENSEN 18 T/M 65 JAAR = ', MENS
      STOP
      END
```

We noemen deze versie PROG2B.F77. Uitvoering geeft het volgende beeld te zien:

```
OK, RUN PROG2B
GEEF LEEFTIJD: 16
GEEF LEEFTIJD: 25
GEEF LEEFTIJD: 31
GEEF LEEFTIJD: 12
GEEF LEEFTIJD: 28
GEEF LEEFTIJD: 69
GEEF LEEFTIJD: -1
 AANTAL MENSEN 18 T/M 65 JAAR =           3
**** STOP

OK,
```

1. Zie *Subroutine Reference Guide*, DOC3621-190L.

4.6 TERMINAL IN- EN UITVOER MET FORMAT-BESTURING

Tot nu toe hebben we in dit hoofdstuk uitsluitend de 'READ *' - en de 'PRINT *' -opdrachten toegepast. Deze opdrachten worden in *Cursus FORTRAN 77* voor bijna alle in- en uitvoer gebruikt. Ze hebben bij de Prime betrekking op de terminal, en werken onder zogenaamde *lijstbesturing*. Dat wil zeggen de 'lijst' na de komma bepaalt hoe en met welk (standaard) format wordt gelezen of geschreven.

Behalve een lijstbestuurde vorm hebben deze opdrachten ook een formatbestuurde vorm (zie hoofdstuk 10 van *Cursus FORTRAN 77*). Daarbij wordt het sterretje eenvoudig vervangen door een directe of indirecte FORMAT-specificatie, bijvoorbeeld:

```
      READ 4, NAAM, ADRES, CODE
4     FORMAT (A30, A42, I7)
```

Hierin is de 4 na de READ een label die naar de daaropvolgende FORMAT-opdracht verwijst. In dit geval hebben we met een indirecte FORMAT-specificatie te maken. Deze indirecte vorm is de enige die in *Cursus FORTRAN 77* ter sprake komt. Omdat de directe specificatie ook handig kan zijn, zullen we daar nu kort bij stilstaan.

Bij de directe vorm wordt de formatspecificatie rechtstreeks in de READ of PRINT opgenomen, en wel in de vorm van een CHARACTER-constante of -variabele. In plaats van de bovenstaande constructie krijgen we dan:

```
READ '(A30, A42, I7)', NAAM, ADRES, CODE
```

of:

```
CHARACTER*14 OPMAAK
OPMAAK = '(A30, A42, I7)'
READ OPMAAK, NAAM, ADRES, CODE
```

Vooral de laatste vorm biedt perspectieven. Deze kan namelijk nuttig zijn in gevallen waarbij de formatgegevens afhankelijk zijn van de programmaresultaten. De formatspecificatie kan dan tijdens programmaexecutie worden bepaald, bijvoorbeeld door de formatstring zelf op te laten bouwen of te kiezen, of door de formatstring samen met de invoergegevens in te lezen. We spreken hierbij van *run-time format*.

Tenslotte nog een voorbeeld van een directe formatspecificatie:

```
 PRINT '(A8, I4, A13, F12.4)',
$      'Som van ', N, ' getallen is ', SOM
```

Is N = 5 en SOM = 120.25, dan wordt hierbij afgedrukt:

```
Som_van____5_getallen_is_____120.2500
```

waarbij _ een spatie voorstelt.

Net als de lijstbestuurde READ en PRINT hebben de formatbestuurde READ en PRINT betrekking op de eigen terminal.

4.7 IN- EN UITVOER MET ANDERE DEVICES

Voor de in- en uitvoer van programmagegevens is tot nu toe uitsluitend gebruik gemaakt van het standaard in- en uitvoerapparaat (*device*), te weten de terminal. Dit device wordt automatisch 'aangesproken' bij uitvoering van een "READ *", "READ n", "PRINT *", of "PRINT n" opdracht. In- en uitvoer via andere devices, bijvoorbeeld de printer (regeldrukker) of het achtergrondgeheugen, is uiteraard ook mogelijk. Een programma zou bijvoorbeeld gegevens kunnen inlezen van een bestand dat eerder met de editor of met een of ander toepassingsprogramma is aangemaakt, en dat zich in het achtergrondgeheugen bevindt.

Om met andere devices dan de terminal te kunnen werken moet gebruik worden gemaakt van in- en uitvoeropdrachten die buiten de SF/k subset vallen. Vanwege hun belang zullen we ze hier kort behandelen.

4.7.1 De uitgebreide sequentiële READ-opdracht

Naast de bekende "READ *" of "READ n" opdracht kan men in FORTRAN 77 ook gebruik maken van een uitgebreidere leesopdracht, die (toevallig) eveneens de naam READ heeft. Om misverstanden te voorkomen zullen we deze aanduiden als de uitgebreide sequentiële READ-opdracht.

Deze opdracht heeft de volgende algemene gedaante:

```
READ ([UNIT=]unitnr [[,FMT=]format]
        [,END==label] [,ERR=label] [,IOSTAT=ios])
                                        [invoerlijst]
```

toelichting:

1. *unitnr* is een INTEGER-expressie die aangeeft van welk invoerdevice gelezen moet worden. Elk device wordt namelijk met een symbolisch unitnummer aangegeven. Met zo'n unitnummer wordt in feite een logische verbinding met een bepaald device bedoeld.

Daarbij kunnen verschillende unitnummers naar één en hetzelfde device verwijzen. De relevante unitnummers voor de HBO Prime-systemen zijn vermeld in tabel 4-1. Ze kunnen voor de duur van het programma worden veranderd door middel van de PRIMOS subroutine ATTDEV. Voor meer informatie hierover zie de Prime manual *Subroutine Reference Guide*, publikatienummer DOC3621-190L.

2. *format* is een facultatieve verwijzing naar een FORMAT-opdracht, of een alternatieve FORMAT-specificatie. Die kan bestaan uit:
   - een CHARACTER-array, -arrayelement, -variabele of -constante
   - een CHARACTER-expressie van vaste lengte
   - een asterisk (*).

   In het laatste geval hebben we te maken met een variant op de bekende lijstbestuurde READ. In plaats van een vast invoerdevice (bij de Prime dus een terminal) kan hierbij ook vanaf een ander invoerdevice worden gelezen. We geven daar straks een voorbeeld van.

   Weglaten van FORMAT-verwijzingen of specificaties betekent dat er zonder conversies wordt ingelezen. Meestal zullen de invoergegevens dan eerder met een 'format-loze' WRITE (zie § 4.7.2) naar het invoerdevice (in dit geval het achtergrondgeheugen) zijn geschreven. Door het achterwege blijven van conversies kan zowel het eerder uitgevoerde schrijfproces als het nu aan de orde zijnde inleesproces sneller plaatsvinden. Bovendien is het mogelijk dat voor de gegevensopslag op schijf in niet-geconverteerde vorm minder ruimte nodig is. Dit hangt natuurlijk af van de aard van de desbetreffende gegevens.

3. De specificatie *END=label* kan worden opgegeven om te voorkomen dat de uitvoering van een programma voortijdig door gebrek aan invoergegevens eindigt, bijvoorbeeld als het laatste getal ingelezen is. *label* verwijst naar de programmaopdracht waar de executie in dat geval moet worden voortgezet. Net als de straks te bespreken IOSTAT-optie is de END een fraaier alternatief voor de sluitkaart of -regel besproken in § 5.4 van *Cursus FORTRAN 77*.

4. De ERR-optie werkt op soortgelijke wijze als de END-optie. Er wordt naar de door *label* aangegeven opdracht gesprongen bij het optreden van een transmissiefout tijdens het inlezen.

5. *IOSTAT* is een afkorting van input/output status. Daarbij is *ios* een INTEGER-variabele die na uitvoering van de READ de status van het inleesproces als volgt weergeeft:
   - een positieve waarde betekent dat er een fout is opgetreden
   - de waarde nul betekent dat de READ correct is uitgevoerd
   - een negatieve waarde betekent dat een einde-bestand (end-of-file) conditie is opgetreden, maar zonder verdere consequenties voor de correcte uitvoering van de opdracht (althans vanuit het systeem bezien).

Door middel van een IF-opdracht, waarin op de variabele *ios* wordt getest, kan tijdens de uitvoering van het programma het verloop van het inleesproces worden nagegaan. Situaties zoals het optreden van einde-bestand kunnen daardoor 'programmatisch' worden ondervangen.

6. *invoerlijst* stelt de variabele of variabelen voor waaraan de ingelezen waarden worden toegekend. Weglaten hiervan betekent dat er wel een record wordt ingelezen, maar dat er geen toekenning van de desbetreffende waarden plaatsvindt. In feite wordt er dus een record overgeslagen.

tabel 4.1 : Belangrijke unitnummers met bijbehorende devices. Zowel de in- als uitvoerdevices zijn vermeld.

FORTRAN unitnummer	device		
1	eigen terminal		
5	PRIMOS file unit	1	(achtergrondgeheugen)
6	PRIMOS file unit	2	(achtergrondgeheugen)
7	PRIMOS file unit	3	(achtergrondgeheugen)
.		.	
.		.	
20	PRIMOS file unit	16	(achtergrondgeheugen)
29	PRIMOS file unit	17	(achtergrondgeheugen)
.		.	
.		.	
139	PRIMOS file unit	127	(achtergrondgeheugen)
140	printer unit 0		
141	printer unit 1		

De gegevens die in bovenstaande syntaxdefinitie tussen blokhaken staan zijn facultatief (zie de notatieafspraken aan het begin van dit boek). Ze worden daarom ook wel *opties* genoemd.

Voor veel toepassingen zal een van de volgende (verkorte) vormen van de READ voldoende zijn.

1. READ(UNIT=unitnr, FMT=*) invoerlijst

   of, bij weglating van de facultatieve delen:

   READ(unitnr, *) invoerlijst

voorbeeld: `READ (5, *) A`

Deze vorm is vergelijkbaar met de bekende opdracht:

READ *, invoerlijst

In beide gevallen geldt lijstbesturing. In het laatste geval kan echter uitsluitend via de terminal worden ingelezen.

2. READ(UNIT=unitnr, FMT=label) invoerlijst

of:

READ(UNIT=unitnr, FMT=charactergrootheid) invoerlijst

waarbij eveneens "UNIT=" en "FMT=" weggelaten mogen worden.

voorbeelden:

1.
```
C     VOORBEELD 1: LEZEN MET UNITSPECIFICATIE EN FORMAT LABEL

      READ   (5, 100) A
100   FORMAT (F10.4)
```

2.
```
C     VOORBEELD 2: IDEM MET CHARACTER-GROOTHEID ALS FORMAT

      CHARACTER*7 OPMAAK
      OPMAAK = '(F10.4)'
      READ   (5, OPMAAK) A
```

Beide vormen zijn vergelijkbaar met de opdracht:

READ format, invoerlijst

die in hoofdstuk 10 van *Cursus FORTRAN 77* ter sprake komt. In beide gevallen wordt onder formatbesturing gelezen. De laatstgenoemde vorm geldt in principe echter alleen voor terminalinvoer.

4.7.2 De uitgebreide sequentiële WRITE-opdracht

Met deze opdracht kan onder lijst- of formatbesturing naar een willekeurig device worden geschreven.

De opdracht heeft de volgende algemene gedaante:

```
WRITE ([UNIT=]unitnr [,[FMT=]format]
        [,ERR=label] [,IOSTAT=ios]) uitvoerlijst
```

UNIT, FMT, enz. hebben dezelfde betekenis als bij de uitgebreide sequentiële READ, met dien verstande dat *ios* nooit negatief kan zijn. Bij het schrijven kan namelijk geen end-of-file conditie optreden. Om dezelfde reden ontbreekt de END-optie. Weglaten van de uitvoerlijst betekent dat er een leeg record wordt geschreven. Voor meer informatie over de samenstelling van de opdracht wordt verwezen naar § 4.7.1.

Voor veel toepassingen zal een van de volgende vormen van de uitgebreide sequentiële WRITE voldoende zijn:

1. WRITE (UNIT=unitnr, FMT=*) uitvoerlijst

   of:

   WRITE (unitnr, *) uitvoerlijst

   voorbeeld: `WRITE (5, *) 'RESULTAAT = ', X`

   Deze vorm is vergelijkbaar met de opdracht:

   PRINT *, uitvoerlijst

   Beide opdrachten werken met lijstbesturing. Voor het gegeven voorbeeld betekent dit dat de waarde van X in standaard opmaak verschijnt. De WRITE heeft echter het voordeel dat naar een ander device dan de terminal kan worden weggeschreven.

2. WRITE (UNIT=unitnr, FMT=label) uitvoerlijst

   of:

   WRITE (UNIT=unitnr, FMT=charactergrootheid) uitvoerlijst

   waarbij eveneens "UNIT=" en "FMT=" weggelaten mogen worden.

   voorbeelden:

   1.
```
C     VOORBEELD 1: MET FORMAT-LABEL EN -OPDRACHT

      WRITE  (5, 500) X
500   FORMAT (' ', 'RESULTAAT = ', F12.4)
```

   2.
```
C     VOORBEELD 2: MET CHARACTER-FORMAT

      CHARACTER*26 F500
      F500 = '('' RESULTAAT = '', F12.4)'
      WRITE  (5, F500) X
```

   Deze vorm is vergelijkbaar met de opdracht:

   PRINT format, uitvoerlijst

   die in hoofdstuk 10 van *Cursus FORTRAN 77* ter sprake komt. In beide gevallen wordt onder formatbesturing geschreven. Bij de PRINT kan alleen naar de terminal worden geschreven, bij de WRITE eventueel ook naar een ander device.

4.7.3 De OPEN-opdracht

Om een bestand in het achtergrondgeheugen te kunnen oproepen of aanmaken moeten bepaalde kenmerken daarvan worden opgegeven. Bijvoorbeeld:

- Met welk unitnummer moet het bestand worden gelezen of geschreven?

- Welke naam heeft het bestand?
- Bestaat het bestand al, of moet het worden aangemaakt?

Deze (en eventueel ook andere) kenmerken kan men in FORTRAN 77 met een zogenaamd *OPEN*-statement specificeren. Na een correcte uitvoering van een OPEN-statement is een bestand via de bekende in- en uitvoeropdrachten bereikbaar. De unitspecificatie in deze opdrachten moet uiteraard overeenkomen met de unitspecificatie in de bijbehorende OPEN.

Voor onze doeleinden kan met de volgende vorm van de OPEN-opdracht worden volstaan (1):

```
OPEN ([UNIT=]unitnr, [,FILE=bestandnaam]
          [,STATUS=stat] [,IOSTAT=ios] [,ERR=label])
```

toelichting:

1. [*UNIT*=]unitnr. Hierin is *unit* een 4-byte INTEGER-waarde die het nummer van de file-unit aangeeft (zie tabel 4-1). Het verwijst naar het device waarop het te openen bestand zich bevindt of moet worden aangemaakt.

2. [,*FILE=bestandnaam*] is een facultatieve vermelding van de naam van het te openen bestand. Geeft men dit niet op, dan krijgt het bestand automatisch de naam F#nnn, waarbij "nnn" het nummer is van de desbetreffende file-unit. Aanbevolen wordt invoerbestanden de extensie IN1, IN2, IN3, enz. te geven, en uitvoerbestanden de extensie UIT1, UIT2, UIT3, enz. De bestandnaam wordt opgegeven als een CHARACTER-expressie, bijvoorbeeld:

   ```
   OPEN ( ... , FILE='PROG.IN1', ... )
   ```

   of:

   ```
   CHARACTER*8 NAAM
   NAAM = 'PROG.IN1'

   ...

   OPEN ( ..., FILE=NAAM, ... )
   ```

3. Met *STATUS* geeft men aan of het om het openen van een reeds bestaand bestand, een nieuw permanent bestand, of een nieuw tijdelijk bestand gaat. De te specificeren waarden zijn respectievelijk 'OLD', 'NEW' en 'SCRATCH'. 'Tijdelijk' wil zeggen: voor de duur van het programma. Is niet bekend of een bestand bestaat, dan kan men STATUS='UNKNOWN' opgeven. Net als bij de FILE-optie kan men in plaats van CHARACTER-constanten (bijvoorbeeld 'OLD') ook CHARACTER-variabelen gebruiken.

---

1. **Voor meer informatie zie hoofdstuk 4 van de Prime publikatie *FORTRAN 77 Reference Guide*, nr. DOC4029-192P.**

4. Bij de optie *IOSTAT=ios* is *ios* een 4-byte INTEGER-variabele die de waarde nul krijgt als het openen van het aangegeven bestand is gelukt.

5. *ERR=label* geeft aan bij welke opdracht de executie moet worden voortgezet als het openen van het bestand mislukt. Het kan in principe worden gebruikt als een alternatief voor de IOSTAT-optie. Voor een toepassingsvoorbeeld zie § 4.7.5.

Voor meer informatie wordt verwezen naar hoofdstuk 4 van de *FORTRAN 77 Reference Guide*, nr. DOC4029-192P.

4.7.4 De CLOSE-opdracht

De CLOSE is de tegenhanger van de OPEN-opdracht. Hij zorgt ervoor dat de koppeling tussen het bestand en de bijbehorende unit wordt verbroken.

De CLOSE-opdracht ziet er als volgt uit:

```
CLOSE ( [UNIT=]unitnr [,STATUS=stat]
                 [,ERR=label] [,IOSTAT=ios] )
```

Afgezien van *STATUS=stat* hebben alle delen van de opdracht dezelfde betekenis als bij OPEN. Voor *stat* kan men KEEP of DELETE invullen. KEEP (= bewaar) betekent dat het bestand na afsluiting door CLOSE blijft voortbestaan. DELETE (= verwijder) betekent dat het bestand na de CLOSE verdwijnt. Voor een bestand dat met STATUS='SCRATCH' is geopend kan bij een CLOSE geen STATUS='KEEP' worden opgegeven.

opmerkingen:

1. De opties kunnen (zoals gebruikelijk) in willekeurige volgorde worden gespecificeerd.

2. Bij een normale beëindiging van een programma worden alle gegevensbestanden die door dat programma zijn geopend automatisch 'geCLOSEd'. Breekt het programma voortijdig af, dan blijven alle geopende bestanden open.

4.7.5 Toepassingsvoorbeeld

In deze paragraaf passen we PROG2 zo aan, dat alle in- en uitvoer van en naar bestanden op het achtergrondgeheugen verloopt. Daarbij komen de belangrijkste facetten van de in § 4.7 behandelde stof aan de orde. We noemen het nieuwe programma PROG2D.F77.

```
C     AANPASSING PROG2 VOOR IN- EN UITVOER VAN/NAAR BESTANDEN IN
C     HET ACHTERGRONDGEHEUGEN

      INTEGER MENS, LFTIJD

      OPEN (UNIT=5, FILE='PROG2D.IN1',  STATUS='OLD', ERR=40)
      OPEN (UNIT=6, FILE='PROG2D.UIT1', STATUS='NEW', ERR=50)

      READ (5 ,*) LFTIJD
      MENS = 0

20    IF (LFTIJD .EQ. -1) GO TO 30
         IF (LFTIJD .GE. 18 .AND. LFTIJD .LE. 65) THEN
            MENS = MENS + 1
         END IF
         READ (5, *) LFTIJD
         GOTO 20
30    CONTINUE

      WRITE (6, *) 'AANTAL MENSEN 18 T/M 65 JAAR = ', MENS

      CLOSE (UNIT=5)
      CLOSE (UNIT=6)

      STOP

40    PRINT *, 'OPENEN BESTAND PROG2D.IN1 MISLUKT'
      PRINT *, 'PROGRAMMA VOORTIJDIG BEEINDIGD'
      STOP

50    PRINT *, 'OPENEN BESTAND PROG2D.UIT1 MISLUKT'
      PRINT *, 'PROGRAMMA VOORTIJDIG BEEINDIGD'
      STOP

      END
```

opgaven/toetsvragen:

1. Toets programma PROG2D.F77 met behulp van de editor in, en laat het door de computer uitvoeren.

2. Wordt het programma correct uitgevoerd? Zo neen, waarom niet? Neem zonodig maatregelen om het programma correct uitgevoerd te krijgen.

3. Laat het resultaat van het programma zowel op het beeldscherm als op de regeldrukker zien.

APPENDIX 1: BEKNOPT OVERZICHT EDITCOMMANDO'S

In dit appendix wordt in alfabetische volgorde een beknopt overzicht gegeven van alle editcommando's. De belangrijkste basiscommando's zijn behandeld in hoofdstuk 3 van dit boek. Waar dat van toepassing is wordt voor nadere informatie dan ook naar de desbetreffende paragraaf in dit hoofdstuk (of naar andere delen van dit boek) verwezen.

Er wordt in geen geval aanspraak gemaakt op een uitputtende beschrijving van de commando's. Daarvoor zult u de desbetreffende Prime-documentatie (*The New User's Guide to Editor and Runoff*, nr. FDR3104-101A) moeten raadplegen.

A[PPEND] tekst

Heeft tot gevolg dat de actuele regel wordt uitgebreid met *tekst*. In *tekst* zijn puntkomma's normaal gesproken niet toegestaan. Als dit al niet door de systeembeheerder permanent is veranderd, kan deze situatie voor de duur van de editsessie worden veranderd met het SYMBOL-editcommando. Het APPEND-commando is behandeld in § 3.2.7.

BR[IEF]

Zorgt voor onderdrukking van de controle- of verificatie-uitvoer die normaal gesproken na de commando's APPEND, CHANGE, FIND, GMODIFY, LOCATE, MODIFY, NEXT en NFIND verschijnt.

B[OTTOM]

Stelt de line-pointer in op de laatste regel van de werkfile. Zie § 3.1.2.

C[HANGE] /tekst1/tekst2 [G] [n]

Vervangt *tekst1* door *tekst2*. Zie § 3.2.4.

D[ELETE] [n]

Verwijdert n regels, inclusief en vanaf de actuele regel. Zie § 3.2.5.

D[ELETE] TO doeltekst

Verwijdert alle regels vanaf de actuele tot aan de regel waarin *doeltekst* voorkomt. Laatstgenoemde regel wordt zelf niet verwijderd.

DU[NLOAD] fnaam [n]

Afkorting van "delete and unload". Maakt een nieuwe file met de naam *fnaam*, kopieert vanaf de actuele regel *n* regels naar deze nieuwe file, en verwijdert de desbetreffende regels uit de werkfile. Handig dus voor het verschuiven van stukken tekst. Let op: specificatie van een bestaande file heeft tot gevolg dat die file wordt vernietigd!

DU[NLOAD] fnaam TO doeltekst

Variant van DUNLOAD. In plaats van een vast aantal regels worden alle regels vanaf de actuele t/m de regel waarin *doeltekst* voorkomt naar de nieuwe file gekopieerd en uit de werkfile verwijderd.

E[RASE] x

Verandert voor de duur van de editsessie (of tot een eventueel volgend ERASE-commando) het zogenaamde ERASE-teken (zie § 1.2). Standaard is dit de CTRL-H combinatie). Door de systeembeheerder kan een ander standaard ERASE-teken zijn ingesteld.

FIL[E] [fnaam]

Zorgt voor het opbergen van de werkfile in het achtergrondgeheugen. Zie § 3.1.6 en 3.2.8.

F[IND] doeltekst

Stelt de line-pointer in op de eerstvolgende regel na de actuele die met *doeltekst* begint. Zie § 3.2.2.

F[IND] (n) doeltekst

Variant van bovengenoemde FIND. Er kan worden aangegeven dat de doeltekst in kolom n begint.

G[MODIFY]

Voert wijzigingen uit op de actuele regel op een karakter-voor-karakter / kolom-voor-kolom basis.

INPUT { (ASR) (PTR) (TTY) }

Leest tekst vanaf een teletype ponsbandlezer (ASR), high-speed ponsbandlezer (PTR) of terminal (TTY, = verstekwaarde). Doorgaans niet relevant voor HBO-systemen.

I[NSERT] tekst

Voegt na de actuele regel *tekst* als nieuwe regel van de werkfile toe. Zie § 3.1.4.

K[ILL] karakter

Stelt een ander KILL-teken in. Het reeds ingestelde KILL-teken is te achterhalen met het PSYMBOL-editcommando. Zie ook § 1.2.

LI[NESZ] n

Verandert de regellengte van de werkfile. Verstekwaarde (tevens maximum) is 1024.

LOA[D] fnaam

Kopieert de inhoud van de file *fnaam* naar de werkfile, te beginnen na de actuele regel. Zie ook UNLOAD.

L[OCATE] doeltekst

Stelt de pointer in op de eerstvolgende regel na de actuele waarin *doeltekst* voorkomt. Zie § 3.2.2.

MODE { CKPAR NCKPAR }

Maakt controle op pariteit van tekens mogelijk. CKPAR heeft tot gevolg dat alle tekens met pariteitsbit O in oktale vorm (^nnn) worden afgedrukt. NCKPAR (= verstekwaarde) zorgt dat tekens normaal worden afgedrukt, ongeacht de waarde van het pariteitsbit. Het pariteitsbit is standaard 1.

MODE { COUNT [n1] [n2] [n3] vorm NCOUNT }

Definieert een teller met startwaarde *n1*, stapgrootte *n2* en veldbreedte *n3*, waarvan de waarde in de plaats van een telsymbool wordt gesubstitueerd. Geldt bij gebruik van de commando's APPEND, INSERT, OVERLAY, RETYPE en GMODIFY. *vorm* kan zijn: PRINT (drukt voorloopnullen af), SUPPRESS (drukt geen voorloopnullen af), of BLANK (vervangt voorloopnullen door spa-

ties). Het telsymbool is standaard @, maar kan door het SYMBOL-editcommando worden gewijzigd. NCOUNT (= verstekwaarde) geeft het telsymbool voor normaal gebruik vrij.

MODE { NUMBER | NNUMBER }

NUMBER heeft tot gevolg dat bij het afdrukken van de werkfile met PRINT, en bij controle-/verificatie-uitvoer, een regelvolgnummer vooraan de regel verschijnt. Het nummer maakt geen deel uit van de eigenlijke werkfile. NNUMBER (= verstekwaarde) schakelt dit mechanisme uit. Kan nuttig zijn in samenhang met het WHERE-editcommando.

MODE { COLUMN | NCOLUMN }

Na MODE COLUMN verschijnt bij iedere overgang van edit- naar inputmode een genummerd regelbeeld op het scherm. Handig bij het intoetsen van FORTRAN-programma's en tabellen. NCOLUMN (= verstekwaarde) schakelt MODE COLUMN (weer) uit.

MODE INFO

Geeft de mogelijkheid met RETURN/ENTER regel voor regel door een werkfile te 'stappen'. Geldt alleen voor recente versies van de editor. Zie § 3.1.2.

MODE { PROMPT | NPROMPT }

MODE PROMPT zorgt voor het afdrukken van een promptteken: & voor input- en $ voor editmode. Andere prompttekens kunnen desgewenst met het SYMBOL-editcommando worden ingesteld. NPROMPT (= verstekwaarde) schakelt MODE PROMPT weer uit.

MODE { PRALL | PRUPPER | PRLOWER }

Geeft de mogelijkheid om op een terminal uitsluitend met hoofdletters of kleine letters te werken.

M[ODIFY] /tekst1/tekst2/ [G] [n]

Werkt als CHANGE, maar tast de oorspronkelijke regel-/kolom-indeling niet aan. Handig bij het doorvoeren van tekstwijzigingen van ongelijke lengte bij tabellen etc.

MOV[E] buffer2 { buffer1 | /tekst/ }

Brengt *tekst* of een tekstregel van *buffer1* naar *buffer2*. De standaard buffers zijn EDLIN (= de regel die wordt ingetoetst), INLIN (= de actuele regel), STR.1 (= vrije buffer, synoniem STRA), STR.2 (= vrije buffer, synoniem STRB), STR.3 (= vrije buffer, synoniem STRC) en STR.4 t/m STR.10 (= vrije buffers, geen synoniemen). Wordt gebruikt in samenhang met het XEQ-editcommando (zie verderop).

N[EXT] [n]

Verhoogt (n>0) of verlaagt (n<0) de waarde van de regelpointer. Zie § 3.1.2.

NF[IND] tekst

Variant van FIND. Werkt precies omgekeerd, dat wil zeggen de pointer wordt ingesteld op de eerstvolgende regel na de actuele die niet met *tekst* begint.

OOPS

Alleen beschikbaar op recentere versies van de editor. Doet het effect van het voorgaande commando teniet, althans voor zover dat commando slechts één regel heeft gemodificeerd of verwijderd. Handig bij (kleine) vergissingen, vandaar de naam ...

OUT[PUT] { (DISPLAY) | (TTY) }

DISPLAY biedt de mogelijkheid om controle-uitvoer in plaats van op de eigen terminal op een speciale terminal te laten afdrukken. Doorgaans niet relevant voor HBO-systemen. Verstekwaarde TTY (= eigen terminal).

O[VERLAY] tekst

Met dit commando kan *tekst* als het ware over de inhoud van de actuele regel worden 'gelegd'. Dat wil zeggen ieder teken van de actuele regel wordt vervangen door het teken op de overeenkomstige plaats in *tekst*. Vervanging van een teken vindt niet plaats als het desbetreffende teken in *tekst* een zg. WILD-teken is (verstekwaarde !).

PA[USE]

Door een PAUSE kan de editor tijdelijk worden verlaten, zonder gevolgen voor de inhoud van de werkfile en de stand van de pointer. Na het verlaten van de editor komt men op PRIMOS-niveau terecht. Met een LISTF-commando, bijvoorbeeld, kan men dan con-

troleren welke files reeds aanwezig zijn, waardoor vergissingen met dubbele filenamen bij het FILEn voorkomen kunnen worden. Ook andere toepassingen zijn denkbaar. Men komt weer terug in de editor door het (PRIMOS-)commando START te geven.

PO[INT] n

Positioneert de pointer op regel *n*. Wordt meestal in samenhang gebruikt met WHERE (voor incidentele gevallen) of MODE NUMBER (als men min of meer doorlopend met genummerde tekstregels werkt). Zie § 3.1.2.

P[RINT] [n]

Drukt *n* regels vanaf de actuele regel af. Zie § 3.1.5.

PS[YMBOL]

Geeft een overzicht van de tekens die een speciale betekenis hebben voor de editor. Zie ook beschrijving van het editcommando SYMBOL.

PT[ABSET] tab1, tab2 ...

Afkorting van *p*hysical *t*abset. Dit commando 'vertelt' de editor hoe de fysieke tabulatorstops van de gebruikte terminal zijn ingesteld. Daardoor kan hij in voorkomende gevallen volstaan met het genereren van tabulatiecodes in plaats van spatiereeksen.

PU[NCH] { (ASR) / (PTP) } [n]

Ponst *n* regels op een teletypeponser (ASR), of op een high-speed ponsbandmachine (PTP). Doorgaans niet van toepassing op HBO-systemen.

Q[UIT]

Hiermee verlaat men de editor zonder de werkfile te bewaren. Zie § 3.1.6.

R[ETYPE] tekst

Vervangt de actuele regel door *tekst*. Zie § 3.2.3.

S[YMBOL] func sym

Verandert een voor de editor 'gereserveerd' symbool, dat wil zeggen een symbool dat door de editor als een speciale functie wordt geïnterpreteerd (zie bijvoorbeeld de beschrijving van het KILL-

teken, § 1.2). De standaard symbolen zijn samengevat in de volgende tabel.

func	sym	betekenis
BLANKS	#	'masker'-teken voor spatie (LOCATE)
COUNTE	@	telsymbool bij MODE COUNT
CPROMP	$	promptteken voor editmode bij MODE PROMPT
DPROMP	&	idem inputmode
ERASE	CTRL-H	verwijdert voorgaand teken (oktaal 210)
ESCAPE	`	voorkomt uitvoering van de bijzondere functie die een speciaal symbool heeft
KILL	?	verwijdert actuele regel
SEMICO	|	scheidingsteken voor editcommando's
TAB	\	sprong naar volgende tabpositie
WILD	!	maskerteken (FIND, NFIND, OVERLAY)

De met SYMBOL ingestelde wijzigingen gelden slechts voor de duur van de editsessie. Zie ook het PSYMBOL-editcommando.

TA[BSET] tab1, tab2, tab3 ... tab8

Met dit commando kunnen tabulatorinstellingen worden vastgesteld. De tabsprongen zelf worden geactiveerd door het gebruik van het "\" teken (backslash). Zie § 1.2.

T[OP]

Dit commando stelt de pointer in op de zogenaamde NULL-regel aan het begin van de werkfile. Zie § 3.1.2.

U[NLOAD] fnaam [n]

Kopieert bij positieve *n* regels in 'voorwaartse' richting (dat wil zeggen naar het einde van de werkfile) naar een nieuwe file *fnaam*. Is n negatief dan worden vanaf de actuele regel *n* regels in 'terugwaartse' richting (dat wil zeggen naar het begin van de werkfile) gekopieerd. Let op: mocht er reeds een file met de naam *fnaam* aanwezig zijn, dan wordt deze overschreven. Verstekwaarde n = 1.

U[NLOAD] fnaam TO doeltekst

Variant van UNLOAD. In plaats van een vast aantal regels worden alle regels, vanaf de actuele tot aan de regel waarin *doeltekst* voorkomt, gekopieerd. Tussen TO en *doeltekst* dient een spatie te staan.

V[ERIFY]

Dit commando is normaal in werking. Het heeft betrekking op het automatisch afdrukken van een gewijzigde actuele regel ten gevolge van een APPEND-, CHANGE-, FIND-, GMODIFY-, LOCATE-, MODIFY-, NEXT-, NFIND-, OVERLAY- of POINT-commando. Deze controle- of verificatie-uitvoer kan worden onderdrukt met het BRIEF-editcommando.

W[HERE]

Drukt het nummer af van de actuele regel. Zie ook POINT.

X[EQ] buffer

Dit commando gaat er van uit dat *buffer* een geheugengebied is met als inhoud een commandoregel, en voert dit (die) commando('s) uit. *buffer* wordt gevuld met het MOVE-commando (zie boven). Commando's kunnen zoals gebruikelijk in één commandoregel worden gecombineerd door ze te scheiden met puntkomma's. De MOVE/XEQ-combinatie is een feite een 'macro' of 'subroutine' faciliteit. Eerst wordt namelijk een verzameling opdrachten gedefinieerd (MOVE). Daarna kan deze opdrachtenverzameling worden geactiveerd (XEQ). Handig bij het herhaald uitvoeren van een of meer commando's.

* [n]

Heeft tot gevolg dat het voorgaande commando *n* keer (of tot het bereiken van begin of einde van de werkfile) wordt uitgevoerd.

ANTWOORDEN OP DE OPGAVEN EN TOETSVRAGEN

§ 3.1.5, blz.17

Neen. De editor 'herkent' uitsluitend zijn 'eigen' commando's. SLIST is een algemeen commando op PRIMOS-niveau (zoals LOGIN, ED, etc.) en valt niet onder de editorcommando's.

§ 3.2.1, blz.20

De editor gaat direct in EDIT-mode. Dit in tegenstelling tot een nieuwe file waar in het begin alleen INPUT-mode zinvol is.

§ 3.2.2, blz.22

1. Ja. Commando: LOCATE X. Dit commando heeft hetzelfde effect als FIND X, omdat vóór de gezochte regel geen regel voorkomt die met X begint.

2. FIND X heeft hetzelfde effect als bij het eerste gebruik van dit commando, d.w.z. de pointer wordt ingesteld op regel XXXXXXXX. LOCATE X zou als resultaat de regel AAAAAAAX opleveren, LOCATE XX de regel XXXXXXXX.

§ 4.5, blz. 34 (tweede opdracht)

```
C     AANPASSING PROG2 MET INVOERPROMPTS D.M.V. PRINT

      INTEGER MENS, LFTIJD

      PRINT *, 'GEEF LEEFTIJD: '
      READ *, LFTIJD
      MENS = 0

20    IF (LFTIJD .EQ. -1) GO TO 30
         IF (LFTIJD .GE. 18 .AND. LFTIJD .LE. 65) THEN
            MENS = MENS + 1
         END IF
         PRINT *, 'GEEF LEEFTIJD: '
         READ *, LFTIJD
         GOTO 20
30    CONTINUE

      PRINT *, 'AANTAL MENSEN 18 T/M 65 JAAR = ', MENS
      STOP
      END
```

Het nadeel van dit programma is dat invoerwaarden niet direct achter de bijbehorende prompt kunnen worden ingetoetst. Iets technischer uitgedrukt: er wordt een ongewenste carriage return/line feed uitgevoerd.

§ 4.7, blz.44

2. Het programma eindigt voortijdig als het invoerbestand PROG2D.IN1 ontbreekt. Dit bestand kan met de editor worden aangemaakt.

3. SLIST PROG2D.UIT
   SPOOL PROG2D.UIT

GERAADPLEEGDE LITERATUUR

1. *Aanvulling op Cursus Pascal voor gebruikers van PRIME computers met HBO-faciliteiten*
   A. van der Sluis en C.A.C. Görts
   Academic Service, Den Haag, 1981
   ISBN 90 6233 084 3

2. *Cursus FORTRAN 77*
   J.N.P. Hume en R.C. Holt (vertaling J.M. den Haan)
   Academic Service, Den Haag, 1981
   ISBN 90 6233 052 5

3. *New User's Guide to EDITOR and RUNOFF*
   Daniel P. Dern
   Prime Computer, Inc., Framingham, Massachusetts 01701, 1980
   FDR3104-101B

4. *Subroutine Reference Guide*
   Prime Computer, Inc., Framingham, Massachusetts 01701
   DOC3621-190L

5. *The FORTRAN 77 Reference Guide*
   Prime Computer, Inc., Framingham, Massachusetts 01701, 1980
   IDR4029

INDEX

ACADEMIC SERVICE INFORMATICA UITGAVEN

INLEIDINGEN

Computers en onze samenleving van M.A. Arbib

Basiskennis informatieverwerking van Jan Everink

De viewdata revolutie van S. Fedida en R. Malik

De informatiemaatschappij van Jan Everink

Informatica, een theoretische inleiding van dr. L.P.J. Groenewegen en prof.dr. A. Ollongren

AIV, Automatisering van de informatieverzorging van ir. Th.J. Derksen, drs. H.W. Crins en drs. L.B. Essink

Organisatie, informatie en computers van David M. Kroenke

MICROCOMPUTERS

Programmeercursus Microsoft BASIC van Nok van Veen

Werken met bestanden in BASIC van L. Finkel en J.R. Brown

Werken met bestanden in Apple-BASIC van L. Finkel en J.R. Brown

Werken met Visicalc van C. Klitzner en M.J. Plociak, Jr.

CP/M: een gids voor zelfstudie van J.N. Fernandez en R. Ashley

Cursus Z-80 assembleertaal van Roger Hutty

Exidy sorcerer en BASIC van Nok van Veen e.a.

TRS-80 BASIC: een gids voor zelfstudie van B. Albrecht e.a.

TRS-80 BASIC voor gevorderden van Don Inman e.a.

Ontdek de ZX-Spectrum van Tim Hartnell

PROGRAMMEREN/PROGRAMMEERTALEN

Inleiding tot het programmeren, deel 1 van ir. J.J. van Amstel e.a.

Inleiding tot het programmeren, deel 2 van ir. J.J. van Amstel e.a.

JSP-Jackson struktureel programmeren van Henk Jansen

Aspecten van programmeertalen van ir. J.J. van Amstel en ir. J.A.A.M. Poirters

Programmeertalen, een inleiding van ir. J.J. van Amstel e.a.

Het Groot Pascal Spreukenboek van H.F. Ledgard , P.A. Nagin en J.F. Hueras

Cursus Pascal van prof.dr. A. van der Sluis en drs. C.A.C. Görts

Cursus eenvoudig Pascal van prof.dr. A. van der Sluis en C.A.C. Görts

Inleiding programmeren in Pascal van C. van de Wijgaart

BASIC, EIT-serie, deel 3

Cursus BASIC van ir. R. Bloothoofd e.a.

Cursus COBOL van dr. A. Parkin

Struktuur en stijl in COBOL van ir. E. Dürr en dr.ir. F. Mulder

Cursus FORTRAN 77 van J.N.P. Hume en R.C. Holt

Programmeren in LISP van prof.dr. L.L. Steels

Cursus ALGOL 60 van prof.dr. A. van der Sluis en drs. C.A.C. Görts

Programmeren, deel 2: Van analyse tot algoritme van prof.drs. C. Bron

Inleiding programmeren en programmeertechnieken, EIT-serie, deel 1

Inleiding programmeren van prof.dr. R.J. Lunbeck

SYSTEEMPROGRAMMATUUR

Bedrijfssystemen, EIT-serie, deel 4

Systeemprogrammatuur van drs. H. Alblas

Vertalerbouw van drs. H. Alblas e.a.

BESTANDSORGANISATIE/DATABASES

Informatiestructuren, bestandsorganisatie en bestandsontwerp, EIT-serie, deel 5

Gegevensstructuren van R. Engmann e.a.

Bestandsorganisatie van prof.dr. R.J. Lunbeck en drs. F. Remmen

Databases van drs. F. Remmen

INFORMATIEANALYSE/SYSTEEMONTWERP

Voorbereiding van computertoepassingen van prof.dr. A.B. Frielink

Simulatie, een moderne methode van onderzoek van drs. S.K.T. Boersma en ir. T. Hoenderkamp

Systeemontwikkeling volgens S.D.M. van H.B. Eilers

Een samenvatting van de System Development Methodology SDM van PANDATA

Cases op het gebied van administratieve organisatie en informatieverzorging (inclusief systeemontwerp) van prof.dr. P.G. Bosch en H.A. te Rijdt

Uitwerkingenboek bij Cases van prof.dr. P.G. Bosch en H.A. te Rijdt

Gegevensanalyse van R.P. Langerhorst

Evaluation of methods and techniques for the analysis, design and implementation of information systems, editors: J. Blank en M.J. Krijger

Analyse van informatiebehoeften en de inhoudsbeschrijving van een databank van prof.dr. P.G. Bosch en ir. H.M. Heemskerk

Eerlijk en helder van prof.dr. P.G. Bosch

THEORETISCH/COMPUTERSCHAAK/TOEPASSINGEN

Abstracte automaten en grammatica's van prof.dr. A. Ollongren en ir. Th.P. van der Weide

De tekstmachine van dr. M. Boot en drs. H. Koppelaar

Computerschaak, schaakwereld en kunstmatige intelligentie van dr. H.J. van den Herik

Lineaire programmering als hulpmiddel bij de besluitvorming van dr. S.W. Douma

Simulatie en sociale systemen, redaktie: J.L.A. Geurts en J.H.L. Oud

Onderneming en overheid in systeem-dynamisch perspectief, redaktie: A.F.G. Hanken en J.H.L. Oud

Over dit boek

Programmeren is bij uitstek een praktische vaardigheid die zich niet laat aanleren door het bestuderen van boeken. Praktische oefening is dus een onmisbaar deel van het leerproces. Behalve de universiteiten en hogescholen hebben daarom nu ook de meeste instellingen voor middelbaar en hoger beroepsonderwijs de beschikking over eigen computerfaciliteiten in de een of andere vorm.

Voor tientallen HBO-instellingen in Nederland bestaan die faciliteiten uit een PRIME-installatie werkend onder het PRIMOS bedrijfssysteem. Deze *Aanvulling* op het eerder uitgegeven *Cursus FORTRAN 77* is bedoeld als een handreiking aan de student-gebruikers van deze installatie. Het behandelt o.a. de algemene communicatie met het systeem, het opbouwen en modificeren van tekstfiles en het verwerken van programma's. Ook het aansluiten van files komt aan de orde.

Kortom: het boek helpt de student over de drempel bij het in praktijk brengen van de leerstof van *Cursus FORTRAN 77*.